Titre: Title:	Simulation d'une tâche d'insertion à l'aide d'un robot manipulateur
Auteur: Author:	Étienne Lachance
Date:	2004
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Lachance, É. (2004). Simulation d'une tâche d'insertion à l'aide d'un robot manipulateur [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7322/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7322/
Directeurs de recherche: Advisors:	
Programme: Program:	Unspecified

UNIVERSITÉ DE MONTRÉAL

SIMULATION D'UNE TÂCHE D'INSERTION À L'AIDE D'UN ROBOT MANIPULATEUR

ETIENNE LACHANCE
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES (M.Sc.A.)
(GÉNIE ÉLECTRIQUE)
AVRIL 2004

Your file Votre référence
ISBN: 0-612-91951-X
Our file Notre référence
ISBN: 0-612-91951-X

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

SIMULATION D'UNE TÂCHE D'INSERTION À L'AIDE D'UN ROBOT MANIPULATEUR

présenté par : LACHANCE Etienne

en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées

a été dûment accepté par le jury d'examen constitué de :

M. HURTEAU Richard, D.Ing, président
M. GOURDEAU Richard, Ph.D., membre et directeur de recherche
M. BIGRAS Pascal, Ph.D., membre

À Audrey, longue vie remplie de joie.

REMERCIEMENTS

Je voudrais d'abord remercier les membres du jury pour avoir accepté d'évaluer ce mémoire de maîtrise.

Je remercie M. Richard Gourdeau, mon directeur de recherche, pour toute l'aide apportée dans le cadre de ce projet ainsi que pour les conseils et remarques pertinents qu'il m'a prodigués.

Je remercie CAE Inc, plus particulièrement Eric Bouthillier, Sophie Bathlon et Frederick Tran-Khanh qui m'ont permis de travailler à temps partiel.

Je remercie également mes parents, Robert et Lyane, pour les maintes relectures de cet ouvrage et pour leur support.

Enfin, je suis particulièrement reconnaissant à ma conjointe Rachel. Sans son support moral et sa compréhension ce projet n'aurait pu voir le jour.

RÉSUMÉ

Dans cette étude, nous nous intéressons à la simulation d'une tâche d'insertion exécutée par un robot manipulateur *Puma 560*. L'objet à insérer, ainsi que le trou devant le recevoir, sont des cylindres. Le robot simulé doit être aussi économique que possible.

Un contrôleur d'impédance guide le robot, tant dans l'espace libre que dans l'espace contraint. La logique floue est utilisée pour faire varier la rigidité de l'impédance du contrôleur afin qu'il soit mieux adapté aux diverses phases de travail. Par exemple, une faible rigidité dans l'espace contraint permet de minimiser les efforts de contact. On ajuste également l'amortissement par logique floue afin d'éviter des oscillations lors du passage de l'espace libre à l'espace contraint. Un observateur des vitesses angulaires accompagne le contrôleur pour réduire les coûts du robot et obtenir des signaux moins bruités. Une démonstration de la stabilité obtenue grâce au tandem contrôleur observateur est présentée.

Les trajectoires à suivre sont fournies par des splines cubiques paramétriques pour les coordonnées de translation et par des splines de quaternions pour les coordonnées d'orientation.

N'étant pas équipé d'un système de vision, le robot lors de l'insertion doit rechercher le trou en aveugle. Une technique utilisant un tracé en spirale à été utilisée.

Une animation, faite avec *OpenGL*, de la tâche à été réalisée pour faciliter l'analyse. L'animation comprend le robot ainsi que son milieu de travail. Il est à noter que le robot représenté est générique permettant de créer facilement une autre animation.

La librairie *ROBOOP* a été utilisée pour la simulation du robot. La majorité des nouvelles fonctions nécessaires à cette étude on été incorporées à la librairie *ROBOOP*.

ABSTRACT

The goal of this study is to simulate a *Puma 560* robot inserting an object into a hole. The object to be inserted and the receiving hole are cylindrical in shape. The simulated robot must be as economical as possible.

An impedance controller guides the robot, both in free space as in constrained space. Fuzzy logic is used to modify the stiffness of the impedance controller to better adapt it to the many phases of it's work. For example, when in constrained space, low stiffness permits to minimise efforts upon contacting the boundries of the space. Fuzzy logic is also used to adjust the damping in order to avoid oscillations when moving from a free space to a constrained one. An angular speed observer, within the controller, reduces the robot's costs and helps reduce the signals' noise. A demonstration of the stability obtained by the 'controller - observator' is given.

The trajectories for the translation coordinates are given by parametric cubic splines, while quarternion splines are usee for the space coordinates.

Having no system of vision, the robot must search for the hole in a blind manner. A method using a spiral movement has been used.

A computer animation, implemented with *OpenGL*, is provided to facilitate the analysis of the tasks performed by the robot. This animation shows the robot in it's work environment. A generic robot has been used to facilitate other animations.

The *ROBOOP* library has been used to simulate the robot. *ROBOOP* has been enhanced with most of the new functions required by this study.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES NOTATIONS ET DES SYMBOLES

Notations :

P,	position et orientation de l'effecteur.
$q(i)$,	position angulaire du joint i dans le repère i.
q,	vecteur de position angulaire.
$\hat{q}$,	estimé de q.
q,	quaternion unitaire.
s,	partie scalaire du quaternion q.
v,	partie vectorielle du quaternion q.
${}^{i}_{i+1}R$,	matrice de rotation du repère $i+1$ vers le repère i.
$[V\times]$,	matrice de produit croisé du vecteur V.
B,	matrice de tenseur d'inertie.
C,	matrice d'effets de Corriolis et centrifuge.
D,	matrice des effets visqueux.
τ,	couple.
J,	matrice Jacobienne.
R_h	rayon du trou.
R_p	rayon de la tige.
Ph_w	position du trou dans le repère w.
Pp_w	position de la tige dans le repère w.
A_M	plus grande valeur propre de la matrice A.

A_m	plus petite valeur propre de la matrice A.
$\overline{\sigma}(A)$	plus grande valeur propre de la matrice A.
$\underline{\sigma}(A)$	plus petite valeur singulière de la matrice A.

Abréviations :

CLIK	Closed Loop Inverse Kinematics.
CTM	Computed Torque Method.
DLS	Moindres carrés amortis (Damped Least-Squares).
FLS	Fuzzy Logic System.
FLC	Fuzzy Logic Controller.
PD	Proportionnel Dérivé.
CTM	Compute Torque Method.
RCC	(Remote Center Compliance).
ROBOOP	Robotics Object Oriented Package in C++.
Lerp	Linear Quaternions Interpolation.
Slerp	Spherical Linear Quaternions Interpolation.
Squad	Spherical Spline Quaternions Interpolation.
SVD	Singular Value Decomposition.
TP	Très Petit.
P	Petit.
PP	Plutôt Petit.
MP	Moyennement Petit.

MG	Moyennement Grand.
PG	Plutôt Grand.
G	Grand.
TG	Très Grand.

Constantes physiques :

$g = 9.81$ m.s^{-2} , accélération de la gravité.

LISTE DES ANNEXES

CHAPITRE 1

INTRODUCTION

L'assemblage de pièces par des robots manipulateurs est une tâche ayant beaucoup d'applications dans l'industrie. La tâche d'insertion (*peg-in-hole*) peut être considérée comme un des problèmes classiques de la robotique. Ce problème a été étudié dans plusieurs recherches pour tenter de trouver une solution générale.

Une tâche d'insertion, ou d'assemblage, est facilement réalisable par un humain, par contre il en est autrement lorsqu'elle doit être exécutée par un robot manipulateur. Ce phénomène est en grande partie expliqué par le manque de compliance de la part du poignet et du bras du robot.

La compliance est l'habileté d'adapter continuellement le mouvement pour obtenir les perceptions désirées. L'assemblage automatique est efficace lorsque la compliance adapte le mouvement du manipulateur. Le contrôle automatique de la compliance peut être défini comme étant la correction du mouvement planifié en utilisant l'information de la mesure de force de contact avec l'environnement.

Un robot, bien entendu n'est pas conscient de ce qu'il fait. Il bouge l'outil et l'oriente pour ensuite effectuer une certaine tâche comme prendre un objet, visser, peinturer, etc.

Dans le problème de l'insertion d'un objet, si celui-ci n'est pas bien aggripé par l'outil du robot ou si le trou n'est pas à l'endroit prévu, alors une intervention humaine est généralement requise. Ceci est dû à des imprécisions dans la description du milieu

de travail.

1.1 Notre travail

Cette étude se penche sur la problématique de la simulation d'une tâche d'insertion exécutée par un robot manipulateur. L'objet à insérer et le trou sont de forme cylindrique. Nous ne nous intéressons pas aux insertions de tiges multiples.

Le robot simulé devrait coûter le moins cher possible. C'est pour cette raison qu'il ne possédera pas de système de vision artificielle ni de capteur de vitesse angulaire des joints.

Le présent mémoire est structuré de la façon suivante : Le chapitre 2 présente une revue de la littérature traitant de l'insertion de pièces par un robot manipulateur et de divers contrôleurs compliants. Le chapitre 3 présente principalement la modélisation cinématique et dynamique d'un robot manipulateur selon la notation *Denavit-Hartenberg* modifiée. La présentation d'une interface graphique animant une tâche complète d'insertion, réalisée à l'aide d'*OpenGL* [1], accompagne la modélisation. La description de l'environnement du robot est faite au chapitre 4. On y montre comment obtenir les différents points de contact entre le trou et la tige (objet à insérer dans le trou) dans le cas particulier où les deux sont de forme cylindrique. On retrouve également une section traitant d'un algorithme utilisé pour simuler de façon approximative un impact entre l'effecteur du robot et l'environnement. La modélisation du robot et de l'environnement étant établie, le chapitre 5 se concentre sur le contrôleur

[1] www.sgi.com/software/opengl/

d'impédance et sur un observateur des vitesses angulaires des joints. Le chapitre 6 traite de techniques utilisées pour accomplir une tâche d'insertion : la modification des paramètres du contrôleur d'impédance par logique floue, la génération de trajectoires et la recherche aveugle d'un trou. Finalement le chapitre 7 présente les résultats de nos simulations.

1.2 Contributions

Cette étude nous a permis d'apporter les contributions suivantes :

1. L'amélioration de la librairie *ROBOOP* [25]. Toutes les simulations ont été faites avec cette dernière.
2. L'élaboration d'un modèle graphique générique pour robot manipulateur en *OpenGL*. Ce modèle est également disponible sous ROBOOP.
3. La modélisation des points de contacts entre une tige cylindrique et un trou cylindrique.
4. Une preuve de la stabilité de la combinaison d'un observateur de vitesses angulaires des joints et d'un contrôleur de position basé sur l'accélération résolue.
5. Une preuve de la stabilité de la combinaison d'un observateur de vitesses angulaires des joints et d'un contrôleur de position basé sur la méthode du couple précalculé.
6. L'utilisation de la logique floue pour ajuster certains paramètres du contrôleur d'impédance.

CHAPITRE 2

REVUE DE LITTÉRATURE

2.1 Compliance

Le contrôle de l'interaction d'un robot manipulateur avec son environnement est essentiel dans l'exécution de plusieurs tâches où l'effecteur doit manipuler un autre objet ou tout simplement faire certaines actions sur une surface. Durant l'interaction, l'environnement génère des contraintes sur les trajectoires pouvant être suivies par l'effecteur. On parle alors de mouvement contraint ou d'espace contraint. L'utilisation d'un contrôleur de mouvement dans l'espace contraint est voué à l'échec. Le succès d'une tâche d'interaction utilisant un contrôleur de mouvement peut être obtenue seulement si on peut planifier avec précision la tâche en question. Pour ce faire il est nécessaire d'avoir un modèle précis du robot (cinématique et dynamique) ainsi que celui de l'environnement. Il est possible d'obtenir un modèle assez précis d'un robot en utilisant des techniques d'identification [1]. Par contre il en est autrement pour le modèle de l'environnement. En pratique, les erreurs de planification peuvent engendrer des forces de contact faisant dévier l'effecteur de la trajectoire désirée. De son côté le contrôleur de mouvement réagit à ces déviations de position pour tenter de réduire l'écart entre la position actuelle et celle désirée. Cette situation fait en sorte que les forces de contact augmenteront jusqu'à la saturation des actuateurs ou jusqu'au bris mécanique d'une pièce. On pallie à cet inconvénient en utilisant un contrôleur ayant

un comportement compliant.

La compliance est nécessaire chaque fois que l'effecteur du robot est contraint par la géométrie de la tâche. Le principale problème relié à la compliance est d'en spécifier les caractéristiques pour une tâche donnée [35]. Deux possibilités s'offrent pour représenter la forme de la compliance : 1) déterminer si un degré de liberté donné est contrôlé en position ou en force, 2) obtenir la matrice de rigidité qui établit une relation linéaire entre la force (ou couple) de l'effecteur et le déplacement (translation ou rotation) de l'effecteur [2].

La compliance sur le mouvement de l'effecteur peut se retrouver sous forme passive ou sous forme active. La compliance passive est la tendance inhérente d'un mécanisme à s'adapter à la force appliquée sur lui en bougeant dans une certaine direction. La compliance passive peut se diviser en deux catégories. Dans la première [51], le désalignement est compensé par le déplacement élastique de pièces du système. Dans la deuxième, qui semble moins utilisée que la première, l'erreur d'alignement est corrigée par l'application de forces et/ou couples de façon aléatoire ou prévue.

Dans la première catégorie, on installe sur l'outil un dispositif mécanique composé de ressorts et d'amortisseurs, que le robot maintient dans une certaine orientation. On a montré que le problème d'insertion peut être facilité si les rigidités latérales et de rotation sont faibles dans l'outil. Le dispositif RCC (*Remote Center Compliance*) est basé sur ce principe [51]. Le dispositif RCC utilise le fait qu'il est plus facile de tirer un objet d'un trou que de l'enfoncer dans le trou. Une force appliquée au centre de compliance causera une translation pure, tandis qu'un moment appliqué à ce point

causera une rotation pure autour de ce point. Les dispositifs passifs, comme le RCC, sont peux dispendieux, cependant ils ne permettent de résoudre que certains problèmes précis. Ils permettent uniquement de s'adapter aux faibles erreurs d'alignement de position entre les pièces. De plus, ils sont sensibles aux changements de l'environnement de l'assemblage.

Contrairement à la compliance passive, la compliance active peut s'appliquer à plusieurs situations variées. Par contre la compliance active a le désavantage de rendre le contrôleur beaucoup plus complexe. L'utilisation de capteurs permet d'obtenir des informations qui seront ensuite utilisées dans le contrôle du système. Les projets réalisés dans ce domaine peuvent être classés selon les capteurs utilisés : la vision artificielle, les capteurs de proximité et les capteurs de force. Un bref résumé des contrôleurs actifs utilisant les capteurs de force est présenté à la section 2.2.

2.2 Contrôle actif de la force de contact

Les techniques actives de contrôle de l'interaction peuvent être groupées en deux catégories : celles effectuant un contrôle indirect de la force et celles effectuant un contrôle direct de la force. La différence entre les deux catégories est que la première fait un contrôle de la force par l'asservissement du mouvement, sans rétroaction sur la force ; la deuxième asservit la force à une valeur déterminée.

Dans la première catégorie, on retrouve le contrôle de la rigidité (*stiffness control*) et le contrôle d'impédance. Le contrôle de la rigidité est la stratégie de base pour le contrôle de la compliance. Il s'agit souvent d'un contrôle proportionel-dérivé (dans

l'espace cartésien) avec compensation de la gravité. La rigidité apparente du robot est contrôlée, ce qui fait que le contrôleur agit comme un ressort de rigidité variable [51]. De façn similaire, le contrôle de l'amortissement (*damping control*) change l'amortissement apparent du manipulateur [51].

Pour sa part, le contrôle d'impédance (*impedance control*) est une généralisation du contrôle de la rigidité et du contrôle de l'amortissement. Il s'agit d'imposer une certaine dynamique à l'effecteur du robot par une impédance mécanique [28], ce qui permet de contrôler l'interaction entre le robot et l'environnement. Ce contrôleur reste stable même lorsque le mouvement de l'effecteur passe de l'espace libre à l'espace contraint [29]. L'effecteur se comporte alors comme un système masse-ressort-amortisseur. Pour une force donnée la position varie en fonction de l'impédance choisie. Il n'y a donc pas de contrôle explicite de la force. Il n'est pas nécessaire de faire la cinématique inverse du robot puisque le contrôleur est exprimé dans l'espace opérationnel [28]. Une des difficultés de la mise en oeuvre est de déterminer une impédance acceptable. Comme toutes les techniques de contrôle avancées, on doit connaître le modèle du robot pour obtenir de bonnes performances. L'obtention d'un bon modèle peut se faire par des techniques d'identification des paramètres inertiels [1].

Si un modèle précis de l'environnement est disponible alors il est possible d'utiliser un contrôle hybride force/position. Cette technique contrôle la position dans l'espace libre et contrôle la force dans l'espace contraint. Par une matrice de décision, on permet de choisir quels degrés de liberté seront contrôlés en position et lesquels seront contrôlés en force. Un degré de liberté ne peut évidemment pas être contrôlé à la fois en position

et en force. Ce contrôleur est sensible aux imprécisions des données géométriques de l'environnement car il doit reconnaître le passage de l'espace libre à l'espace contraint (par la matrice de décision). C'est pour cette raison qu'il ne doit pas être utilisé lorsque la connaissance de l'environnement n'est pas adéquate [16].

Le contrôle parallèle force/position est une nouvelle génération du contrôleur hybride force/position [13]. Deux contrôleurs agissent en parallèle pour contrôler la position et la force. Pour s'assurer que le contrôleur parallèle est compliant, le contrôle en force domine le contrôle en position en cas de contact avec l'environnement.

2.3 Observateur des vitesses angulaires

Plusieurs stratégies permettent de résoudre le problème de régulation et de poursuite de trajectoires. L'ingrédient principal dans la majorité d'entre elles est un contrôleur PD (proportionnel dérivée), ce qui montre que ces techniques reposent sur l'hypothèse de la connaissance des variables d'état q et $\dot{q}$, où q est le vecteur de position angulaire des joints. Malheureusement en pratique cette hypothèse n'est pas vérifiée. Premièrement même si les robots sont généralement pourvus de capteurs de positions de haute précision, les mesures de vitesses, elles, sont effectuées par des tachymètres contaminés par le bruit. Deuxièmement, on omet souvent les capteurs de vitesses pour réduire les coûts, le volume et le poids [5].

On peut obtenir une mesure de la vitesse sans utiliser de tachymètre par une dérivation numérique de premier ordre du signal de position. La simplicité de cette technique la rend attrayante du point de vue de l'implantation. Elle n'est pas conseillée

puisqu'elle génère des signaux bruités comme ceux obtenus à l'aide de tachymètres.

C'est pour ces raisons que plusieurs contrôleurs utilisant un observateur [1] ont été proposés (ex : [5], [41], [26]). Cependant la dynamique complexe d'un robot manipulateur rend la conception d'observateur plus difficile. De plus, rien ne garantit que certains contrôleurs seront toujours stables lors du remplacement des vitesses angulaires obtenues à l'aide de tachymètres par celles obtenues par un observateur donné.

Nicosia et Tomei [41] ont élaboré un observateur non-linéaire. Ils ont prouvé la stabilité locale de l'agencement de leur observateur et de deux contrôleurs différents ; un PD pour la régulation et un contrôle par couple précalculé (*CTM : compute torque method*) modifié pour le suivi de trajectoire.

Berghuis et Nijmeijer ont proposé deux observateurs linéaires accompagnés de deux contrôleurs. Le premier ensemble, assure la régulation de façon globale à l'aide d'un contrôleur PD [5], alors que dans le second ensemble, le contrôleur utilise l'erreur d'observation pour le suivi de trajectoire [42]. Les résultats théoriques du second sont similaires à ceux retrouvés dans [41].

2.4 Insertion

Whitney [51] a décrit les équations quasi-statiques de l'insertion d'une tige cylindrique dans un trou cylindrique. Il fournit des inégalités permettant de détecter les conditions de coincements (*jamming*) et de coincements élastiques (*wedging*). Le coin-

[1] Désormais nous emploirons le terme observateur pour désigner un estimateur des vitesses angulaires.

cement et le coincement élastique sont définis au chapitre 4. Shahinpoor et Zohoor [44] fournissent les équations d'insertion pour le cas cylindrique en régime dynamique, contrairement à la plupart des études qui traitent la situation d'insertion en régime quasi-statique, c'est-à-dire en négligeant les termes d'inertie et de gravité. De plus, ils ont montré qu'une insertion réussie respecte un ensemble d'inégalités.

L'identification des points de contact lors d'une insertion permet de déterminer les équations de mouvement de l'insertion. Jusqu'à ce jour, il y a eu principalement deux types d'analyse de points de contact : celle de l'insertion d'un cylindre dans un trou cylindrique et celle de l'insertion d'un prisme dans un trou prismatique. Contrairement au cas cylindrique qui peut se traiter en deux dimensions sous certaines conditions, le cas prismatique est beaucoup plus complexe. Sturges [49] fournit une étude exhaustive des points de contacts pour le cas prismatique.

Beaucoup d'analyses traitent le problème d'insertion d'une tige cylindrique dans un trou de forme cylindrique seulement en deux dimensions [44]. Ceci revient à dire que les axes longitudinaux de la tige et du trou se coupent. Cette approximation est valide lorsque la différence entre le diamètre du trou et de la tige est assez faible. Bazerghi et Goldenberg [4] ont proposé un algorithme pour déterminer les points de contacts entre une tige cylindrique et un de trou de forme cylindrique en trois dimensions. Leur solution est assez complexe.

Lorsque la position du trou est incertaine, on doit faire appel à des techniques de recherche. Nous n'abordons pas les techniques de vision artificielle puisque nous voulons simuler un robot aveugle. Pettinaro [43], Chhatpar et Branicky [12] font l'analyse

de différents parcours de recherche. Il semble que celui en spirale est le meilleur car il optimise l'aire de recherche et est plus simple à mettre en oeuvre.

2.5 Solution envisagée

1. Modéliser la dynamique de contact entre une tige cylindrique et un trou cylindrique ;
2. Simuler des tâches d'insertion d'une tige rigide dans un trou rigide. L'extrémité de la tige n'est pas biseautée et le trou n'est pas chanfrainé ;
3. Implémanter un contrôleur d'impédance actif tant dans l'espace libre que dans l'espace contraint ;
4. Utiliser l'observateur présenté par Nicosia et Tomei [41] ;
5. Modifier certains paramètres du contrôleur d'impédance en utilisant la logique floue pour obtenir de meilleures performances dans l'espace libre et l'espace contraint ;
6. Modéliser une technique de recherche du trou ne nécessitant pas la vision artificielle ;
7. Utiliser la librairie *ROBOOP* [25] pour nos simulations ;
8. Développer un modèle graphique du robot ainsi que son environnement de travail pour faciliter l'analyse des résultats.
9. Étendre la librairie ROOBOOP avec les éléments de programmation utilisés pour la simulation.

CHAPITRE 3

MODÉLISATION DU ROBOT

Pour faciliter notre tâche d'analyse nous avons décidé d'utiliser une librairie existante modélisant la cinématique et la dynamique d'un robot manipulateur. Deux choix s'offraient, soit la librairie *ROBOOP* [25] écrite en C++ ou la librairie *Robotics Toolbox, for Matlab* [18]. Nous avons choisi la première sous un système d'exploitation Linux. Elle est portable sur plusieurs plateformes et elle est sous une licence GPL.

Tout au long de notre étude nous avons contribué à l'amélioration de la librairie. Les modifications apportées à *ROBOOP* doivent être d'ordre général pour que la librairie demeure applicable à n'importe quel robot manipulateur sériel. *ROBOOP* est disponible sur le site wed *www.cours.polymtl.ca/roboop/*.

ROBOOP est utilisée pour simplifier l'écriture de simulateur de robots manipulateurs (qui sont parfaitement rigides) dans un environnement de programmation similaire à *Matlab* pour la manipulation de matrices.

Nous n'avons pas consacré de section pour la validation de *ROBOOP* étant donné que c'est une librairie stable. On retrouve sous *ROBOOP* un module permettant de la comparer à la librairie *Robotics Toolbox, for Matlab* [18]. La notation utilisée dans ce chapitre est très similaire à celle retrouvée sous *ROBOOP*.

La cinématique et la dynamique d'un robot manipulateur sont présentés aux sections 3.1 et 3.2 respectivement. Les caractéristiques du *Puma 560*, qui est le robot modélisé, sont présentées à la section 3.3. Finalement, la section 3.4 présente une

animation générique d'un robot manipulateur.

3.1 Cinématique du robot manipulateur

Tout au long de l'étude on utilisera deux types de repères, soit le repère cartésien de base et les repères articulaires. Le premier repère, qui est fixe, est lié à l'environnement. Le second type est placé sur chaque joint du robot.

Un robot manipulateur comprend un ensemble de membrures formant une chaîne reliée par des joints. Chaque joint possède un degré de liberté, soit en rotation ou en translation. Dans le premier cas on parlera d'un joint rotoïde et dans le second d'un joint prismatique. Un manipulateur à n joints, numéroté de 1 à n, possède $n+1$ membrures, numérotées de 0 à n. La membure 0 étant la base du manipulateur (fixe dans notre cas) et la membrure n contient l'effecteur. L'effecteur est l'organe terminal du joint n servant à accomplir une tâche. Dans cette étude on emploira effecteur pour désigner le bout de la tige à insérer. Le joint i relie les membrures $i-1$ et i, selon la notation modifiée de Denavit-Hartenberg [20].

a_i la distance entre l'axe z_i et l'axe z_{i+1} mesuré le long de l'axe x_i ;

α_i l'angle entre les axes z_i et z_{i+1} mesuré autour de l'axe x_i ;

d_i la distance entre l'axe x_{i-1} et l'axe x_i mesurée le long de l'axe z_i ;

θ_i l'angle entre les axes x_{i-1} et x_i mesuré autour de l'axe z_i.

a_i est positif puisqu'il s'agit d'une distance ; α_i, d_i, et θ_i sont des quantités signées. Pour un joint rotoïde, θ_i est la variable et d_i est fixe, alors que pour un joint prismatique d_i est la variable et θ_i est fixe.

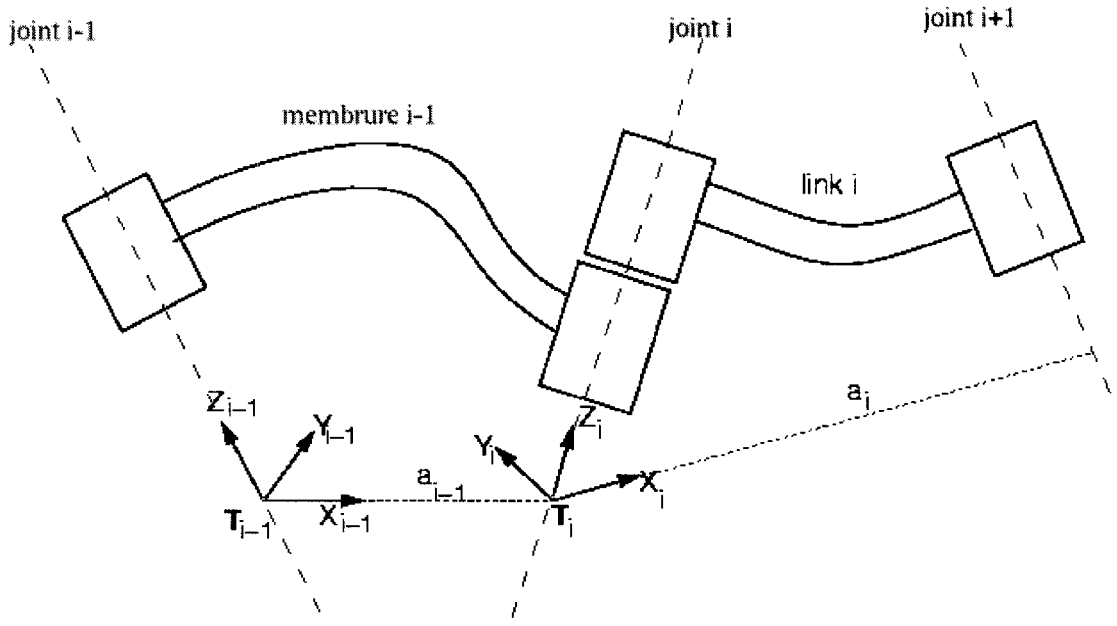


FIG. 3.1 Repère de la notation Denavit-Hartenberg modifiée

Les coordonnées généralisées et les forces généralisées sont représentées par les équations 3.1 et 3.2 respectivement.

$$q_i = \begin{cases} \theta_i & \text{pour un joint rotoïde} \\ d_i & \text{pour un joint prismatique} \end{cases} \tag{3.1}$$

$$Q_i = \begin{cases} \tau_i & \text{pour un joint rotoïde} \\ f_i & \text{pour un joint prismatique} \end{cases} \tag{3.2}$$

La matrice de rotation qui effectue les changements d'axes entre les différents systèmes repères du robot est donnée par l'équation 3.3. La matrice de rotation est orthogonale, ainsi $R^{-1} = R^T$. Il est également possible de représenter une rotation par un quaternion. Il est préférable d'utiliser les matrices pour effectuer les multiplications comparativement aux quaternions, car le nombre d'opérations arithmétiques en est

plus faible. La matrice de transformation homogène 3.5 est utilisée pour obtenir le vecteur position d'un système d'axe par rapport à un autre.

$$ {}^{i-1}{}_{i}R = \begin{bmatrix} c\theta_i & -s\theta_i & 0 \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} \end{bmatrix} \tag{3.3} $$

L'orientation de l'effecteur par rapport au repère de base, ${}^{0}{}_{e}R$, s'obtient en un produit successif de l'équation précédente. La dérivée de cette matrice est reliée à la vitesse angulaire de l'effecteur par la relation suivante

$$ \dot{R}_e = S(\omega_e) R_e \tag{3.4} $$

où $S()$ est la matrice de produit vectoriel tel que définie à l'équation I.19.

La matrice de transformation homogène permet de représenter la position et l'orientation du repère i par rapport au repère $i-1$.

$$ {}^{i-1}_{i}T = \begin{bmatrix} {}^{i-1}{}_{i}R & {}^{i}p \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{3.5} $$

L'inverse de la matrice de transformation est

$$ {}^{i-1}_{i}T^{-1} = \begin{bmatrix} {}^{i-1}{}_{i}R^T & -{}^{i-1}{}_{i}R^T {}^{i}p \\ 0_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} {}^{i-1}{}_{i}R^T & -{}^{i}_{i-1}R^{i}p \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{3.6} $$

Le vecteur ${}^{i}p$ est défini comme étant la position du repère i par rapport au repère $i-1$ exprimé dans le repère $i-1$.

$$ {}^{i}p = \begin{bmatrix} a_i \\ -d_i sin(\alpha_i) \\ d_i cos(\alpha_i) \end{bmatrix} \tag{3.7} $$

La matrice de transformation permet d'obtenir la position de l'effecteur (ou tige) par rapport à la base, pour un vecteur de position des joints donné, comme l'illustre l'équation 3.8.

$$ {}^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \ldots {}^{N-1}_{N}T \tag{3.8} $$

La position et l'orientation de l'effecteur sont reliées au vecteur des joints, q, par $P = f(q)$. De façon similaire les vitesses cartésiennes sont liées au vecteur q et $\dot{q}$ par

$$ \dot{P} = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} = J(q)\dot{q} \tag{3.9} $$

où $J = \frac{\partial f}{\partial q}$ est la matrice Jacobienne. Cette matrice se décompose en autant de vecteurs colonnes que le nombre de joint.

$$ {}^{0}J(q) = \begin{bmatrix} {}^{0}J_1(q) & {}^{0}J_2(q) & \ldots & {}^{0}J_n(q) \end{bmatrix} \tag{3.10} $$

où ${}^0J_i(q)$ est définie comme étant

$$ {}^0J_i(q) = \begin{bmatrix} z_i \times {}^i p_n \\ \\ z_i \end{bmatrix} \quad \text{pour un joint rotoïde} \tag{3.11} $$

$$ {}^0J_i(q) = \begin{bmatrix} z_i \\ \\ 0 \end{bmatrix} \quad \text{pour un joint prismatique} \tag{3.12} $$

Les vecteurs z_i et ${}^i p_n$ représentent respectivement l'axe du joint i exprimé dans le repère i et la position du joint i jusqu'à l'outil exprimé dans le repère i. Ces deux vecteurs sont exprimés dans le repère de base.

$$ {}^iJ(q) = \begin{bmatrix} {}^0_iR^T & 0 \\ \\ 0 & {}^0_iR^T \end{bmatrix} {}^0J(q) \tag{3.13} $$

L'inversion de la Jacobienne permet d'obtenir les vitesses angulaires des joints, $\dot{q}$, pour un vecteur vitesse, v, de l'espace cartésien. Si $J(q)$ est singulière, on peut obtenir une approximation de son inverse par la méthode des moindres carrés amortis. Cette méthode est décrite en annexe II.

La dérivée temporelle de 3.9 fournit la relation entre l'accélération de l'effecteur, a_e, et les vecteurs q, $\dot{q}$ et $\ddot{q}$

$$ a_e = J(q)\ddot{q} + \dot{J}(q, \dot{q})\dot{q} \tag{3.14} $$

dont $\dot{J}(q)$ est donné par les équations 3.15 3.16 et 3.17

$$ {}^0\dot{J}(q) = \begin{bmatrix} {}^0\dot{J}_1(q) & {}^0\dot{J}_2(q) & \cdots & {}^0\dot{J}_n(q) \end{bmatrix} \tag{3.15} $$

$$ {}^0\dot{J}_i(q,\dot{q}) = \begin{bmatrix} \omega_{i-1} \times z_i \\ \omega_{i-1} \times^{i-1} p_n + z_i \times^{i-1} \dot{p}_n \end{bmatrix} \text{ pour un joint rotoïde} \tag{3.16} $$

$$ {}^0\dot{J}_i(q,\dot{q}) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{ pour un joint prismatique} \tag{3.17} $$

La propriété suivante de la dérivée de la Jacobienne

$$ \dot{J}(q,x)y = \dot{J}(q,y)x \tag{3.18} $$

peut être observé par la décomposition du produit $\dot{J}\dot{q}$

$$ \dot{J}\dot{q} = \dot{q}_1 \begin{bmatrix} 0 \\ \dot{u}_1 \end{bmatrix} + \dot{q}_2 \begin{bmatrix} \dot{e}_2 \\ \dot{u}_2 \end{bmatrix} + \cdots + \dot{q}_n \begin{bmatrix} \dot{e}_n \\ \dot{u}_n \end{bmatrix} \tag{3.19} $$

où $\dot{e}_i$ et $\dot{u}_i$ sont les lignes de 3.16. Ces termes représentent les vitesses angulaires et les vitesses linéaires du joint i.

Tout au long du texte on dit que le robot évolue dans l'espace libre lorsque son effecteur peut bouger librement. De la même façon, on dit que le robot évolue dans l'espace contraint lorsqu'il ne peut pas bouger librement, c'est-à-dire que son effec-

teur a perdu au moins un degré de liberté. Par exemple, lors d'une insertion réussie d'un cylindre dans un trou cylindrique, le robot a perdu 4 degrés de liberté, soit les déplacements linéaires x et y et les déplacements angulaires θ_x et θ_y.

3.2 Dynamique du robot

La dynamique d'un robot manipulateur décrit le mouvement du robot en fonction des couples fournis par les actuateurs et par les forces externes appliquées à l'effecteur. Les deux aspects principaux de la dynamique sont la *dynamique inverse* et la *dynamique directe*. Dans le premier, les équations de mouvement du robot sont résolues, pour un mouvement donné, dans le but d'obtenir les forces généralisées, soit les couples aux joints. Pour le second, l'accélération est obtenue pour certains vecteurs de position, de vitesse et de forces généralisées. Nous reviendrons sur ce point à la section 3.2.1.

Le modèle dynamique d'un robot manipulateur peut être écrit sous la forme suivante [20]

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) = \tau - J^T(q)f \tag{3.20}$$

où

q est le vecteur généralisé de coordonnées des joints ;

$\dot{q}$ est le vecteur de vitesse des joints ;

$\ddot{q}$ est le vecteur d'accélération des joints ;

B est la matrice d'inertie symétrique définie positive ;

C décrit les effets des accélérations de Coriolis et centripète ;

D est une matrice diagonale décrivant les effets visqueux et la friction de Coulomb ;

g décrit les effets de la gravité ;

τ est le vecteur des couples actifs aux joints ;

J est la matrice Jacobienne ;

f est le vecteur des forces de contact avec l'environnement.

En montrant comment obtenir l'équation 3.20, nous déduirons quatre propriétés structurales des robots manipulateurs à joints rotoïdes [47] [41] qui seront utilisées lors de la discussion de l'observateur des vitesses angulaires aux sections 5.5 à 5.7. Voici les propriétés

p1 Une définition appropriée de $C(q, \dot{q})$ rend la matrice $\dot{B} - 2C$ anti-symétrique.

p2 Posons deux vecteurs $n \times 1$ x et y on a $C(q, x)y = C(q, y)x$.

p3 La norme de $C(q, \dot{q})$ satisfait l'inégalité $\|C(q, \dot{q})\| \leq k_c \|\dot{q}\|$

p4 La matrice d'inertie possède une borne inférieure positive $\|B^{-1}(q)\| \leq k_2$, avec $k_2 > 0$

p5 Posons deux vecteur $n \times 1$ x et y on a $\dot{J}(q, x)y = \dot{J}(q, y)x$. Cette propriété a été présenté à la section 3.1.

p6 La norme de $\dot{J}(q,\dot{q})$ satisfait l'inégalité $\|\dot{J}(q,\dot{q})\| \leq k_j \|\dot{q}\|$

Notons que les propriétés p2 et p6 sont valables uniquement pour des robots à joints rotoïdes.

Selon la formulation de Lagrange, l'équation de mouvement d'un système peut être obtenue de 3.21, où L est le Langrangien, F est le vecteur des forces généralisées. Rappelons que le Lagrangian est la différence entre l'énergie cinétique et potentielle du système $L = T - U$.

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = F \tag{3.21}$$

$$L = \frac{1}{2}\dot{q}^T B(q)\dot{q} - G(q) \tag{3.22}$$

$$F = -D\dot{q} + \tau - J^T(q) f \tag{3.23}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = \dot{B}(q)\dot{q} + B(q)\ddot{q} \tag{3.24}$$

$$\frac{\partial L}{\partial q} = \frac{1}{2}\dot{q}^T \frac{\partial B(q)}{\partial q}\dot{q} - \frac{\partial G(q)}{\partial q} \tag{3.25}$$

$$g(q) = \frac{\partial G(q)}{\partial q} \tag{3.26}$$

Ainsi l'équation de mouvement est

$$B(q)\ddot{q} + \dot{B}(q)\dot{q} - \frac{1}{2}\dot{q}^T \frac{\partial B(q)}{\partial q}\dot{q} + D(\dot{q}) + g(q) = \tau - J^T(q) f \tag{3.27}$$

Le vecteur $n \times 1$, des forces de Coriolis et centrifuge est donné par

$$C(q,\dot{q})\dot{q} = \dot{B}(q,\dot{q})\dot{q} - \frac{1}{2}\dot{q}^T \frac{\partial B(q)}{\partial q}\dot{q} \tag{3.28}$$

Ce vecteur est une fonction quadratique de $\dot{q}$ et ces éléments peuvent être écrits de la façon suivante

$$C_{ij} = \sum_{k=1}^{n} c_{ijk}\dot{q}_j\dot{q}_k \tag{3.29}$$

Il existe plusieurs choix pour les éléments C_{ij} de la matrice C, correspondant à une factorisation différente de $C(q,\dot{q})\dot{q}$. Le choix

$$c_{ijk} = \frac{1}{2}\Big(\frac{\partial B_{ij}}{\partial q_k} + \frac{\partial B_{ik}}{\partial q_j} - \frac{\partial B_{jk}}{\partial q_i}\Big) \tag{3.30}$$

où c_{ijk} sont les symboles de Christoffel de premier type [47] qui permet de rendre la matrice $\dot{B} - 2C$ anti-symétrique. Cette propriété est utilisée dans le design de l'observateur de la section 5.5.

La propriété p6 peut se démontrer à l'aide de l'équation suivante [22]

$$C(q,\dot{q}) = J^T(q)WMJ(q) + J^T(q)M\dot{J}(q,\dot{q}) \tag{3.31}$$

où W et M représentent la matrice des vitesses angulaires étendues et la matrice des

masses étendues. À l'aide de l'inégalité du triangle on obtient

$$\|C(q,\dot{q})\| \leq \|J\|^2\|W\|\|M\|+\|J\|\|M\|\|\dot{J}(q,\dot{q})\| \tag{3.32}$$

$$k_c\|\dot{q}\| \leq k_p^2 k_w\|\dot{q}\|k_m + k_p k_m\|\dot{J}(q,\dot{q})\| \tag{3.33}$$

On voit maintenant que la norme de la dérivée de la Jacobienne est fonction d'une constante et de la norme du vecteur des vitesses angulaires.

Une simplification au problème d'insertion est que la dernière membrure, l'effecteur ainsi que la tige à insérer sont représentés comme un seul corps. En procédant de cette façon on assume que l'outil de l'effecteur prend la tige parfaitement (sans glissement ni flexion). De cette façon nous ignorons la dynamique entre l'outil et la tige.

3.2.1 Simulation de la dynamique du robot

La méthode de Newton-Euler, qui était déjà implantée sous *ROBOOP*, est utilisée pour simuler la cinématique et la dynamique d'un robot manipulateur. Elle repose sur la somme des forces et moments agissant sur les membrures d'un manipulateur. La solution est récursive, on calcule d'abord la propagation des vitesses et des accélérations de la première membrure jusqu'à la dernière pour ensuite obtenir la propagation des forces et moments de la dernière membrure jusqu'à la première. Cette méthode est très efficace pour une implantation par ordinateur. Les équations de la méthode, respectant la notation Denavit-Hartenberg modifiée, sont illustrées ci-dessous [20].

Définissons quelques termes

- σ_i est le type de joint, $\sigma_i = 1$ pour un joint rotoïde, $\sigma_i = 0$ pour un joint

prismatique ;

- p_i est la position du référentiel i par rapport au référentiel $i-1$;
- r_i est la position du centre de masse de la membrure i ;
- I_{ci} est l'inertie de la membure i par rapport à son centre de masse ;
- I_m est l'inertie de rotation du moteur i ;
- $z_0 = [0\ 0\ 1]^T$;
- Récursion avant pour $i = 1, 2, \ldots, n$.

  Initialisation : $\omega_0 = \dot{\omega}_0 = 0$ and $\dot{v}_0 = -g$.

$$\omega_i = R_i^T \omega_{i-1} + \sigma_i z_0 \dot{\theta}_i \tag{3.34}$$

$$\dot{\omega}_i = R_i^T \dot{\omega}_{i-1} + \sigma_i R_i^T \omega_{i-1} \times z_0 \dot{\theta}_i + \sigma_i z_0 \ddot{\theta}_i \tag{3.35}$$

$$\begin{aligned} \dot{v}_i = {} & R_i^T (\dot{\omega}_{i-1} \times p_i + \omega_{i-1} \times (\omega_{i-1} \times p_i) + \dot{v}_{i-1}) \\ & + (1 - \sigma_i)(2\omega_i \times z_0 \dot{d}_i + z_0 \ddot{d}_i) \end{aligned} \tag{3.36}$$

- Récursion arrière pour $i = n, n-1, \ldots, 1$.

Initialisation : $f_{n+1} = n_{n+1} = 0$.

$$\dot{v}_{ci} = \dot{\omega}_i \times r_i + \omega_i \times (\omega_i \times r_i) + v_i \tag{3.37}$$

$$F_i = m_i \dot{v}_{ci} \tag{3.38}$$

$$N_i = I_{ci}\ddot{\omega}_i + \omega_i \times I_{ci}\omega_i \tag{3.39}$$

$$f_i = R_{i+1} f_{i+1} + F_i \tag{3.40}$$

$$n_i = N_i + R_{i+1} n_{i+1} + r_i \times F_i + p_{i+1} \times R_{i+1} f_{i+1} \tag{3.41}$$

$$\tau_i = \tau_{fi} + \sigma_i n_i z_0 + (1 - \sigma_i) f_i^T z_0 \tag{3.42}$$

$$\tau_{fi} = I_m \ddot{q} + B\dot{q} + C_f sign(\dot{q}) \tag{3.43}$$

On doit utiliser le principe de la dynamique inverse pour simuler le mouvement du robot. On résout d'abord l'accélération, tel qu'illustré par l'équation suivante [20] :

$$\ddot{q} = B^{-1}(q)\Big[\tau - J^T(q)f - C(q,\dot{q})\dot{q} - D\dot{q} - g(q)\Big] \tag{3.44}$$

Sous *ROBOOP* la matrice B^{-1} est obtenue avec la méthode *inertia*.

En utilisant un intégrateur numérique, on calcule les vitesses et positions futures. Nous avons choisi d'utiliser l'intégrateur numérique Runge Kutta d'ordre 4 à pas fixes.

3.3 PUMA 560

Le robot modélisé est le *Puma 560*. L'avantage d'utiliser un robot comme le *Puma 560*, est que l'on connait assez bien ses paramètres cinématiques et dynamiques [19].

Les tableaux 3.1, 3.2 et 3.4 fournissent les paramètres de Denavit-Hartenberg modifiés, les paramètres inertiels du Puma ainsi que les paramètres dynamiques des moteurs respectivement. On peut assumer que les paramètres Denavit-Hartenberg sont connus avec une bonne précision. Il en est autrement pour les paramètres inertiels du robot ainsi que les paramètres des moteurs. Ces valeurs ont été extraites en utilisant des techniques d'identification [19]. Il est difficile d'obtenir les paramètres exacts d'un robot par identification. C'est pour cette raison que les paramètres utilisés par le contrôleur ne seront pas les mêmes que celui du robot. Une erreur allant jusqu'à $\pm 10\%$ a été ajoutée sur les paramètres vus par le contrôleur pour tenter de rendre la simulation un peu plus réaliste. Les tableaux 3.3 et 3.5 représentent les paramètres inertiels du robot et les paramètres dynamiques utilisés par le contrôleur. Les données perturbées peuvent être générées automatiquement en utilisant la fonction *perturb_robot* de *ROBOOP*.

TAB. 3.1 *Paramètres de Denavit-Hartenberg modifiés du Puma 560*

membrure(i)	α_{i-1}	a_{i-1} (m)	d_i (m)	θ_i
1	0	0	0	0
2	$-\pi/2$	0	0	0
3	0	0.4318	-0.1501	0
4	$-\pi/2$	0.0203	0.4318	0
5	$\pi/2$	0	0	0
6	$-\pi/2$	0	0	0

m est la masse d'une membrure, $[k_g]$;

c_i est la position du centre de masse de la membrure selon l'axe i, $[m]$;

I_{ii} est une composante du tenseur d'inertie de la membrure, $[k_g m^2]$.

TAB. 3.2 *Paramètres inertiels du Puma 560*

membrure(i)	m	c_x	c_y	c_z	I_{xx}	I_{xy}	I_{xz}	I_{yy}	I_{yz}	I_{zz}
1	0	0	0	0	0	0	0	0	0	0.35
2	17.4	0.068	0.006	-0.016	0.13	0	0	0.524	0	0.539
3	4.8	0	-0.07	0.014	0.066	0	0	0.0125	0	0.066
4	0.82	0	0	-0.019	0.0018	0	0	0.0018	0	0.0013
5	0.34	0	0	0	0.0003	0	0	0.0003	0	0.0004
6	0.09	0	0	0.032	0.00015	0	0	0.00015	0	0.00004

TAB. 3.3 *Paramètres inertiels du Puma 560 utilisés par le contrôleur*

membrure(i)	m	c_x	c_y	c_z	I_{xx}	I_{xy}	I_{xz}	I_{yy}	I_{yz}	I_{zz}
1	0	0	0	0	0	0	0	0	0	0.35
2	18.0	0.070	0.006	-0.017	0.11	0	0	0.57	0	0.6
3	4.0	0	-0.07	0.016	0.06	0	0	0.01	0	0.075
4	0.80	0	0	-0.02	0.0015	0	0	0.0018	0	0.0013
5	0.30	0	0	0	0.00031	0	0	0.00033	0	0.00045
6	0.09	0	0	0.032	0.0002	0	0	0.0002	0	0.00002

I_m est le moment inertie du rotor selon l'axe de rotation du joint, $[k_g m^2]$;

G_r est le rapport de réduction $[-]$;

B est le coefficient d'amortissement visqueux $[\frac{kgm^2}{s}]$;

C_f est le coefficient de friction de Coulomb $[Nm]$.

3.4 Visualisation à l'aide d'*OpenGL*

Pour faciliter le développement de la simulation, un modèle graphique 3D animé a été créé. Nous obtenons ainsi une visualisation de la simulation. Ce modèle utilise la librairie *OpenGL* [52].

TAB. 3.4 – *Paramètres dynamiques des moteurs du Puma 560*

joint(i)	I_m	G_r	B	C_f
1	190e-6	62.611	1.54e-3	0.445
2	220e-6	107.815	0.82e-3	0.1
3	205e-6	53.706	1.43e-3	0.12
4	34e-6	76.036	78.2e-6	0.014
5	30e-6	71.923	82.6e-6	0.011
6	35e-6	76.686	37.7e-6	0.008

TAB. 3.5 – *Paramètres dynamiques des moteurs du Puma 560 utilisés par le contrôleur*

joint(i)	I_m	G_r	B	C_f
1	190e-6	62.611	1.58e-3	0.4
2	220e-6	107.815	0.82e-3	0.1
3	205e-6	53.706	1.43e-3	0.12
4	36e-6	76.036	70.2e-6	0.014
5	33e-6	71.923	82.6e-6	0.012
6	33e-6	76.686	37.7e-6	0.008

Pour des raisons de simplicité, les membrures et les joints d'un robot sont représentés par des cylindres, ce qui fait que notre méthode de visualisation est générique. En effet il est possible d'animer n'importe quel robot avec la notation *Denavit-Hartenberg*. Les figures 3.2 et 3.3 illustrent deux animations pour des robots différents. La première montre le *Puma 560* alors que la seconde présente le *Schilling Titan II*.

La visualisation d'un robot est faite principalement dans les classes *Robotgl* et *mRobotgl*. La première est consacrée à la notation *Denavit-Hartenberg* alors que la seconde est dédiée à la notation *Denavit-Hartenberg* modifiée. Ces classes se retrouvent sous *GLROBOOP*, qui est une extension de *ROBOOP*. *GLROBOOP* sera bientôt disponible à la même adresse que *ROBOOP*, soit *www.cours.polymtl.ca/roboop/*.

FIG. 3.2 – Animation du Puma 560 à l'aide d'OpenGL.

FIG. 3.3 – Animation du Schilling Titan II à l'aide d'OpenGL.

CHAPITRE 4

MODÉLISATION DE L'ENVIRONNEMENT

Situons l'étude avant de nous attaquer à la modélisation de l'insertion d'une tige dans un trou. Il y a principalement deux grandes catégories d'insertion, soit l'insertion simple ou l'insertion multiple. Par insertion multiple on entend l'insertion de plusieurs tiges dans plusieurs trous à la fois. Nous ne nous intéresserons qu'à la première catégorie. L'insertion simple peut se décomposer en deux classes, soit l'insertion d'une tige cylindrique dans un trou cylindrique et l'insertion qui dépend de l'orientation, comme par exemple l'insertion d'un prisme dans un trou prismatique. Nous ne nous intéresserons qu'à la première pour des raisons de simplicité. Cette classe peut encore se diviser en deux, pour obtenir soit l'insertion dans un seul trou et l'insertion en tandem. Dans l'insertion en tandem, la cavité est formée d'au moins deux trous coaxiaux de diamètres différents. Pour des raisons de simplicité, nous ne nous intéresserons qu'à la première.

La section 4.1 présente une analyse des points de contact entre une tige cylindrique et un trou de forme cylindrique. Pour sa part la section 4.2 décrit la dynamique de l'insertion d'une tige cylindrique dans un trou cylindrique.

4.1 Analyse de contact

Comme première hypothèse de base, nous considérons que la tige est parfaitement rigide et que le trou est modélisé par un ressort. Nous reviendrons sur ce point à la

section 4.2.

Par simplicité, nous assumons que la surface du trou est contenue dans un plan Γ parallèle au plan X_bY_b du repère de base. Ceci n'est pas vraiment une contrainte comme nous le verrons à la section 6.3.2. De plus, l'axe du trou est parallèle à la normale du plan Γ.

Considérons le processus d'insertion entre une tige cylindrique de rayon R_p et un trou de forme cylindrique de rayon R_h. Pour que l'insertion soit possible on doit évidemment avoir $R_h > R_p$.

Après le début de l'insertion, il sera possible d'avoir jusqu'à deux points de contact, mais dans un cas de coaxialité il y aura une infinité de points de contact entre la tige et le trou. Il y a au plus cinq situations différentes de contacts [4] que nous avons considérées, tel qu'illustré à la figure 4.1, soit :

1. Il n'y a pas de contact.
2. Un point de contact existe entre un sommet (point d'un cercle) de la tige et l'intérieur du trou.
3. Un point de contact existe entre la base du trou et la surface de la tige.
4. Il y a contact le long d'une génératrice. On peut considérer un point de contact au milieu du segment de contact, cas B ou C.
5. Deux points de contact sont présents (combinaison de B et C).

Nous proposons une solution simple pour déterminer les points de contact entre la tige et le cylindre en trois dimensions. Notre solution est moins complexe que celle proposée par Bazerghi et Goldenberg [4]. Dans leur étude, il faut faire un calcul d'opi-

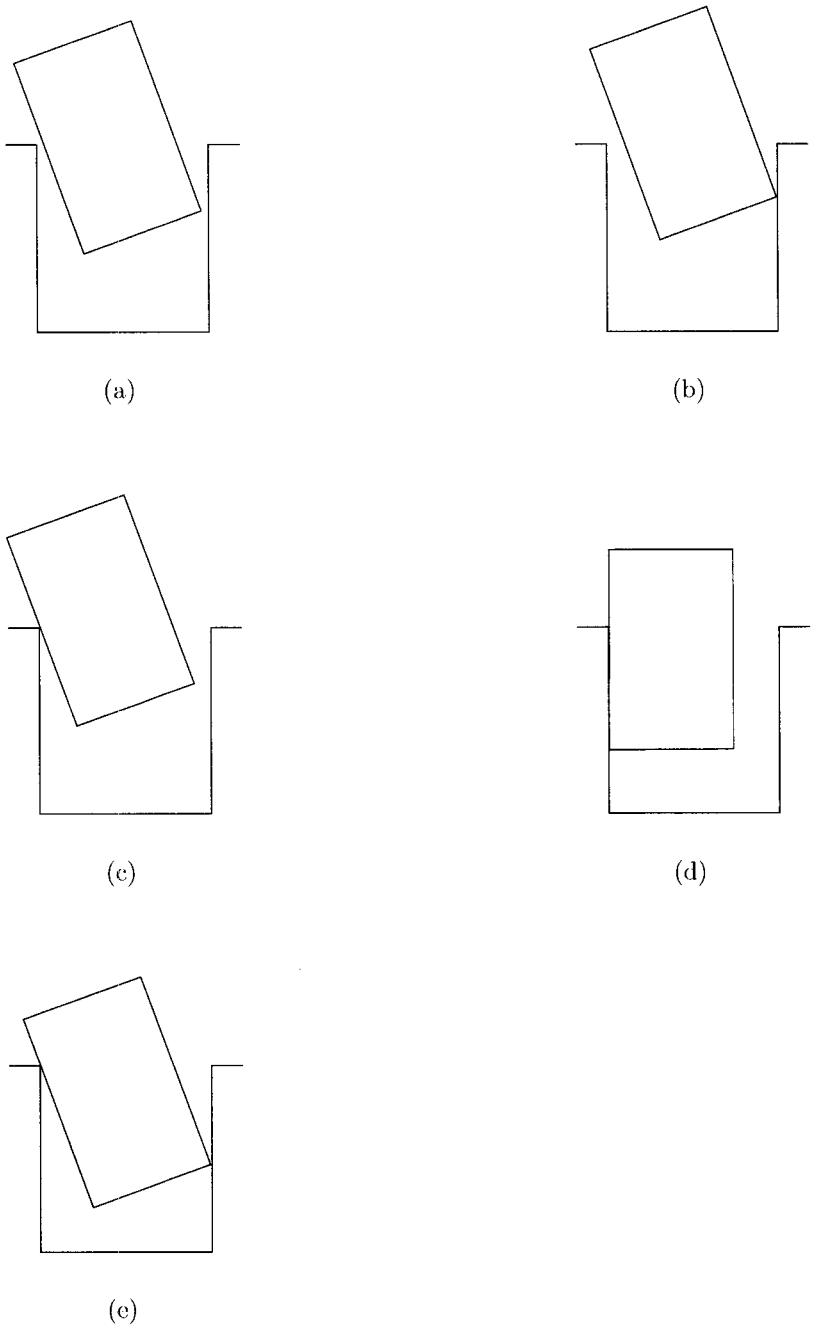


FIG. 4.1 Diagrammes de l'insertion d'une tige cylindrique dans un trou de forme cylindrique. a) sans contact, b) contact entre l'intérieur du trou et le sommet de la tige, c) contact entre la base du trou et la surface de la tige, d) contact le long d'une génératrice et e) contact entre deux points

misation pour déterminer les points de contact lorsque les axes longitudinaux de la tige et du trou ne se croisent pas. Dans notre étude nous décrivons une solution pour déterminer les points de contact en 3 dimensions entre la tige et le trou.

La surface obtenue par l'intersection d'un cylindre et d'un plan, lorsque l'axe longitudinal du cylindre et la normale du plan ne sont pas parallèles, est une ellipse. Le problème revient à déterminer les points de contact d'une ellipse (tige inclinée) contenue dans un cercle (trou), tel qu'illustré à la figure 4.2. Les quatre ellipses (trou incliné) tangentes à chaque quadrant du cercle illustrent quatre positions (parmi une infinité) où il y aura contact entre la tige et le trou. Dans ces quatre positions le centre des ellipses est sur la limite de la zone admissible (en vert). Si le centre de la tige est contenue à l'intérieur de la zone admissible, alors il n'y aura pas de contact entre la tige et le trou. Cette idée fort simple est utilisée pour déterminer les points de contact entre la surface du trou et le sommet de la tige, et entre la base du trou et la surface de la tige.

Les positions, de la tige et du trou, sont exprimées dans le repère X_e, Y_e, Z_e et non pas dans le repère de base X_b, Y_b, Z_b. Ce nouveau repère, illustré à la figure 4.3 est obtenu en effectuant une rotation d'un angle ψ autour de l'axe Z_b. Cela élimine les termes mixtes (xy) dans les équations de points de contact.

Le centre de la tige est inclus dans la zone admissible si l'inégalité suivante est respectée.

$$\frac{(x_e - Ph_{xe})^2}{(R_h - R_p/|\cos\theta|)^2} + \frac{(y_e - Ph_{ye})^2}{(R_h - R_p)^2} \quad < \quad 1 \tag{4.1}$$

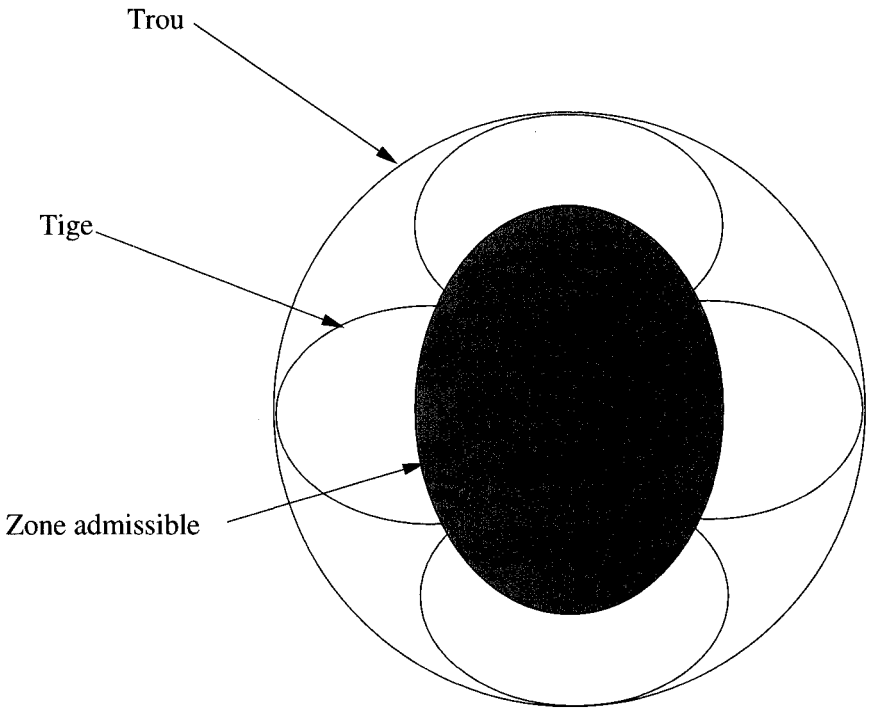


FIG. 4.2 – Zone admise pour l'insertion.

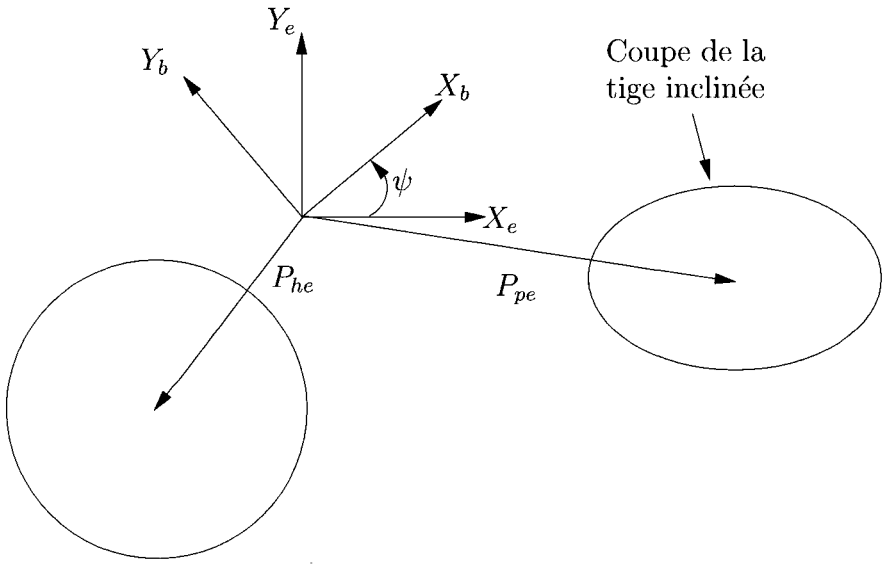


FIG. 4.3 – Repères du trou et de la tige vu en plan. Z_e et Z_b sont parallèles.

où x_e, y_e, Ph_{xe}, Ph_{ye}, R_h, R_p et θ sont respectivement la position x et y du centre de la tige, la position x et y du trou, les rayons du trou et de la tige et l'angle d'inclinaison de la tige par rapport à l'axe Z_e.

Il est assez difficile de résoudre les équations d'un cercle 4.2 et d'une ellipse 4.3 sous forme quadrique pour obtenir les points de contact.

$$(x_e - Ph_{xe})^2 + (y_e - Ph_{ye})^2 = R_h^2 \tag{4.2}$$

$$\frac{(x_e - Pp_{xe})^2}{(R_p/|\cos\theta|)^2} + \frac{(y_e - Pp_{ye})^2}{R_p^2} = 1 \tag{4.3}$$

Il est suffisant de connaître le point de contact lorsque l'ellipse est tangente au cercle, car les autres solutions n'ont pas de sens physique dans notre problème. Nous pouvons déterminer ce point de contact avec les équations paramétriques d'un cercle 4.5 et d'une ellipse 4.6.

$$x_e = Ph_{xe} + R_h\cos(t) \tag{4.4}$$

$$y_e = Ph_{ye} + R_h\sin(t) \tag{4.5}$$

$$x_e = Pp_{xe} + \frac{R_p}{\cos\theta}\cos(t) \tag{4.6}$$

$$y_e = Pp_{ye} + R_p\sin(t) \tag{4.7}$$

En résumé, nous utilisons l'équation 4.1 pour vérifier qu'il y a contact. Lorqu'il y a contact, les équations 4.4 à 4.7 permettent de trouver le point de contact, exprimé dans le repère X_e, Y_e, Z_e entre l'ellipse et le cercle.

Il y a encore un problème pour obtenir le point de contact étant donné que l'environnement est modélisé par un ressort. En effet, nous avons assumé qu'il y a contact lorsque l'ellipse est tangente au cercle (ce qui est conforme à la réalité). Dans notre simulation l'ellipse peut sortir légèrement du cercle à cause de la modélisation du trou par un ressort.

4.2 Dynamique de l'insertion

Deux options s'offraient pour la modélisation de la dynamique entre la tige et le trou. La première est de simuler un robot contraint, alors que la seconde est de simuler l'interaction entre la tige et l'environnement à l'aide d'un ressort.

Les contraintes géométriques, dues à l'environnement, d'un robot manipulateur sont toujours de type holonômes. L'ajout de k contraintes holonômes sur le mouvement réduit le problème de n degrés de liberté à $n - k$ degrés de liberté. Cela consiste à ajouter k équations de contrainte de la forme suivante

$$h_l(q_1, q_2, \ldots, q_n, t) = 0, \quad l = 1 \text{ à } k \tag{4.8}$$

Ceci est une application directe des multiplicateurs de Lagrange. La combinaison des multiplicateurs de Lagrange et de l'équation de mouvement du robot 3.20 donne une matrice d'inertie augmentée qui n'est plus définie positive, parce que les nouvelles lignes et colonnes ajoutées sont linéairement dépendante. Ellis et Ricker [23] ont proposé une méthode pour éliminer les combinaisons linéaires. Ils réussissent à simuler un robot planaire à deux joints dont l'effecteur est contraint dans une rainure. De

plus, plusieurs problèmes d'intégrations numérique ont été observés dans cette étude. Ils ont mentionné que d'autres objectifs de recherche seraient de simuler un robot manipulateur à plusieurs degrés de liberté contraint par une surface.

Comment faut-il aborder le problème lorsque le robot évolue dans l'espace libre pour ensuite évoluer dans l'espace contraint ? L'approche des multiplicateurs de Lagrange nous semble trop complexe pour ce problème. De plus, notre recherche ne porte pas sur la simulation d'un robot contraint par multiplicateurs de Lagrange.

Nous avons décidé d'utiliser la deuxième approche car elle est beaucoup plus simple à mettre en oeuvre. La force normale à l'environnement est donnée par

$$F = k_e \delta x \tag{4.9}$$

où k_e est la rigidité de l'environnement et δx est la pénétration de l'effecteur (ou tige) dans l'environnement.

Cette approche comporte également des problèmes de simulation. Le premier problème est la sélection du pas de calcul. Plus la rigidité k_e est élevée et plus le pas de calcul doit être faible. Par le principe de la dynamique inverse, on obtient l'accélération des joints, que l'on intègre numériquement deux fois pour obtenir la vitesse et la position. Si le pas de calcul est grossier on peut obtenir des variations importantes de positions d'une itération à l'autre. Pour un pas grossier, on risque d'avoir de l'instabilité lors du contact avec l'environnement lorsque la rigidité est élevée. C'est une des raisons pour laquelle nous avons utilisé un faible pas de calcul dans nos simulations.

La figure 4.4 illustre les efforts soumis à la tige par le robot et l'environnement.

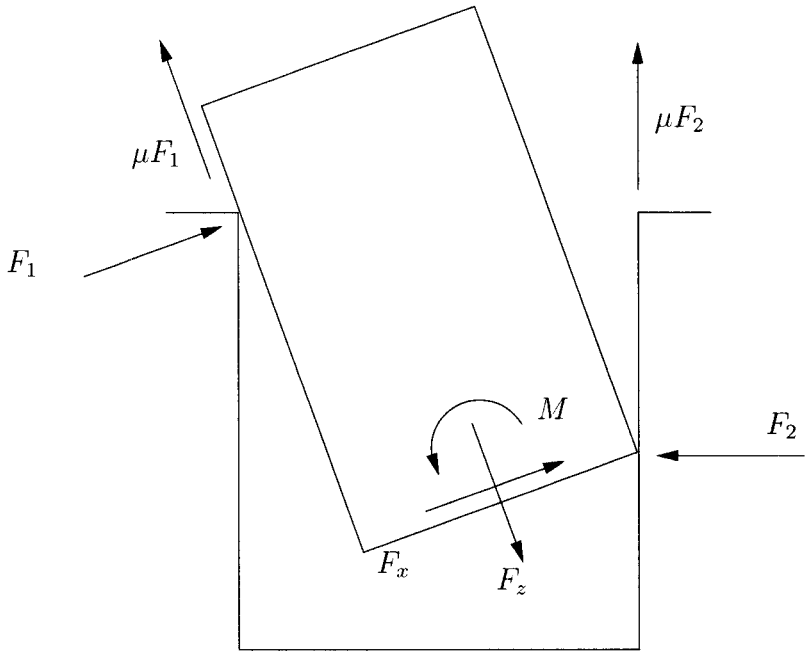


Figure 4.4: Forces de contact entre la tige et le trou.

Notons que nous n'avons pas simulé le phénomène d'impact. Il est possible de prédire les vitesses après impact entre des solides rigides (de formes simples) libres. Il en est autrement lorsqu'on essai de déterminer les vitesses après impact de solides rigides formant une chaîne, comme par exemple un robot manipulateur, et l'environnement. À ce jour, aucune solution ne semble exister pour ce problème.

Pour des raisons de simplicité, le frottement de Coulomb n'est pas simulé. Le frottement de Coulomb est une notion élémentaire de la physique mécanique. Sa simulation pause par contre des problèmes de stabilité, lorsque le corps soumis au frottement a une vitesse approchant zéro. On ne veut pas que le frottement soit une source de

mouvement.

$$F_\mu = F_n \mu \tag{4.10}$$

$$\mu = \begin{cases} \mu_s & \text{si } v = 0 \\ \mu_d & \text{sinon} \end{cases} \tag{4.11}$$

Dans l'équation précédente F_n est la force normale à l'environnement, μ_s est le coefficient de frottement statique et μ_d est le coefficient de frottement dynamique.

Le coincement (jamming) est une condition pour laquelle la tige ne peut pas bouger lorsqu'elle est insérée dans le trou, car les forces et moments appliqués à cette dernière par l'entremise du trou sont dans de mauvaises proportions [51]. Pour éviter cette situation, les forces et moments appliqués par l'outil à la tige doivent être plus grands ou égales aux forces et moments appliqués à la tige par le trou.

Le coincement élastique (wedging) est également une condition pour laquelle la tige ne peut bouger une fois insérée dans le trou. Contrairement au coincement, la cause est géométrique et non liée à des forces et couples qui agissent sur la tige dans de mauvaises proportions. Lorsque le coincement élastique est sévère, aucune force et aucun couple ne peut dégager la tige sans endommager le trou ou la tige aux points de contact [51].

Le coincement et le coincement élastique ne sont pas simulés puisqu'ils dépendent de la friction de Coulomb. Il serait intéressant lors d'une recherche ultérieure de simuler le frottement.

CHAPITRE 5

CONTRÔLE ET OBSERVATEUR

Ce chapitre présente les techniques utilisées pour réaliser une tâche d'insertion. La section 5.1 décrit l'implantation de l'impédance utilisée par un contrôleur de position. La section 5.2 décrit le contrôleur de position appelé contrôle de l'accélération résolue, alors que la section 5.3 décrit le contrôle de position basé sur la cinématique inverse en boucle fermée. La section 5.5 nous renseigne sur un observateur, tandis que les sections 5.6 et 5.7 traitent des tandems observateur contrôleur d'accélération résolue et observateur contrôleur par couple précalculé respectivement.

5.1 Contrôle d'impédance

Parmi les contrôleurs compliants disponibles, le contrôleur d'impédance est le mieux adapté lorsque l'environnement possède à la fois un espace libre et un espace contraint [28]. Il existe principalement deux façons d'implanter un contrôleur d'impédance.

La première utilise une seule boucle de rétroaction pour asservir à la fois la position et la force de contact. La seconde approche, qui est plus robuste que la première, utilise une boucle de rétroaction interne pour la position et une boucle externe pour la force. L'impédance fournit une trajectoire compliante que le contrôleur de position doit suivre [34]. Le contrôleur d'impédance utilisé dans notre recherche, qui est basé sur la deuxième technique, est celui décrit par l'équipe de Siciliano [8] [46].

Il est suffisant de considérer la position de l'effecteur et la force de contact dans l'exécution d'une tâche à trois degrés de liberté. À six degrés de liberté il faut ajouter l'orientation de l'effecteur. L'impédance sera donc formée d'une partie de translation et d'une partie d'orientation.

La représentation minimale la plus connue appelée espace opérationnel, utilise les angles d'Euler pour décrire l'orientation et des coordonnées cartésiennes pour décrire la position. Le premier inconvénient de cette représentation est la possibilité d'avoir des singularités dans la Jacobienne. Le deuxième inconvénient est que la rigidité de rotation du contrôle d'impédance ne peut être décrit directement dans l'espace de travail [46].

La représentation de l'orientation par un vecteur et un angle (angle/axis) plutôt que les angles d'Euler permet d'éviter le deuxième inconvénient. Parmi l'ensemble des descriptions angle/ vecteur seul les quaternions unitaires offrent une représentation sans singularité [31].

L'impédance de translation utilisée est de la forme suivante :

$$M_p\ddot{\tilde{p}} + D_p\dot{\tilde{p}} + K_p\tilde{p} \;=\; f \tag{5.1}$$

Les paramètres M_p, D_p, K_p et $\tilde{p}$ représentant respectivement la matrice d'inertie, la matrice d'amortissement, la matrice de rigidité et la différence entre la position compliante et la position désirée, peuvent être choisis pour rencontrer différents objectifs. Par exemple, on spécifie une rigidité élevée dans les directions où l'environnement est compliant et qu'une bonne poursuite de la position est importante. Inversement, on

spécifie une faible rigidité dans les directions ou l'environnement est rigide ou lorsque des faibles forces de contact doivent être maintenues. On choisit un amortissement élevé lorsqu'on doit dissiper de l'énergie. Finalement, on choisit l'inertie de façon à rendre la réponse de l'effecteur souple lorqu'il est soumis à une force de contact [32].

Une matrice de rigidité équivalente non-diagonale permet de coupler les forces avec des déplacements angulaires, et les moments avec des déplacements linéaires. Ces termes couplés sont utiles pour des tâches avec contact, par exemple éviter des coincements durant des tâches d'insertion complexes [11] [2]. En terme simple, la matrice de rigidité non-diagonale est utilisée pour coupler l'impédance de translation avec l'impédance de rotation. Un mouvement linéaire entraîne une rotation et un mouvement de rotation entraîne un mouvement linéaire. Il n'est pas nécessaire d'utiliser une impédance non-diagonale pour éxécuter une tâche d'insertion simple (cylindre dans un trou cylindrique) [2]. Il en serait autrement si l'orientation de la pièce par rapport à l'axe du trou serait importante (ex : action de visser, insertion d'un prisme dans un trou prismatique). C'est pour cette raison que la matrice de rigidité du contrôleur simulé est diagonale.

Le contrôle d'impédance est implanté en deux phases. La première est la génération de trajectoires compliantes. Ces trajectoires sont obtenues avec les impédances linéaire et de rotation. La deuxième phase utilise la dynamique inverse avec des mesures de forces et de couples pour faire suivre ces trajectoires par l'effecteur. Cette phase est également appelée contrôle selon l'accélération résolue (resolved-acceleration). La figure 5.1 fournit une vue d'ensemble du contrôleur.

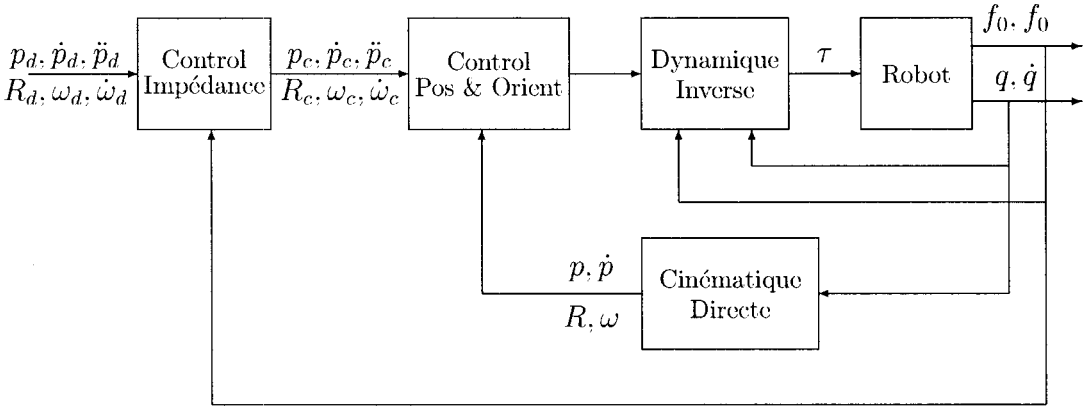


FIG. 5.1 – Schéma bloc du contrôle d'impédance.

L'exposé suivant explique les développements du contrôleur d'impédance.

En robotique on décrit souvent les tâches à éxécuter en termes d'un vecteur de position (3×1), p_d et d'une matrice (3×3) d'orientation, R_d. p_d et R_d représentent l'origine et l'orientation d'un repère désiré par rapport au repère de base. Définissons également p et R comme étant l'origine et l'orientation d'un repère situé sur l'effecteur (ou tige dans la tâche d'insertion) par rapport au repère de base. Lorsque le manipulateur évolue dans l'espace libre, le repère outil doit suivre le repère désiré. Par contre, lorsque le manipulateur est en contact avec l'environnement, il est préférable d'introduire un repère compliant défini par p_c et R_c. En contact avec l'environnement, l'effecteur devra suivre ce repère.

Les erreurs de position, d'orientation et de variation temporelle d'orientation entre

le repère désiré et le repère compliant sont

$$\tilde{p} = p_c - p_d \tag{5.2}$$

$$\tilde{R}^d = R_d^T R_c \tag{5.3}$$

$$\dot{\tilde{R}}_d = S(\tilde{\omega}^d)\tilde{R}^d \tag{5.4}$$

où $\tilde{\omega} = \omega_c - \omega_d$ est l'erreur sur la vitesse angulaire et l'opérateur $S(\cdot)$ est la matrice du produit vectoriel. En utilisant une représentation angle/axe, R^d est exprimé par une rotation de θ autour d'un vecteur unitaire $\tilde{u}$. L'orientation est donc décrite par quatre paramètres (trois pour le vecteur et un pour l'angle).

Nous avons choisi d'utiliser les quaternions unitaires pour exprimer cet ensemble angle/vecteur. L'annexe I et les références [17], [39], [31] et [21] peuvent être consultées afin d'obtenir plus d'information sur les quaternions. Le quaternion calculé depuis l'erreur d'orientation est $\tilde{q} = \{\tilde{s}, \tilde{v}\}$.

Définissons une impédance mécanique à l'effecteur, par une partie de translation et une partie de rotation, basée sur l'énergie. Soit T, et U les contributions de l'énergie cinétique et de l'énergie potentielle. Les termes d'énergie utilisés sont

$$T = \frac{1}{2}\dot{\tilde{p}}^T M_p \dot{\tilde{p}} + \frac{1}{2}\tilde{\omega}^T M_o \tilde{\omega} \tag{5.5}$$

$$U = \frac{1}{2}\tilde{p}^T K_p \tilde{p} + 2\tilde{v}^T K_o \tilde{v} \tag{5.6}$$

où M_p, M_o, K_p et K_o sont des matrices diagonales définies positives. Elles représentent des matrices de masse, de tenseur d'inertie, de rigidités linéaires et de rigidités angu-

laires. Ces matrices sont indépendantes du temps.

$\tilde{v}$ n'étant pas une coordonnée généralisée, nous ne pouvons utiliser la formulation de Lagrande (pour obtenir les équations de mouvement). Les termes des forces et moments conservatifs, des équations d'impédance, sont obtenues en considérant leurs puissances respectives [46]. En prenant les dérivées par rapport au temps des termes d'énergie cinétique, on obtient

$$\dot{T} = \ddot{\tilde{p}}^T M_p \dot{\tilde{p}} + \dot{\tilde{\omega}}^T M_o \dot{\tilde{\omega}} \tag{5.7}$$

où

$$f_i = M_p \ddot{\tilde{p}} \tag{5.8}$$

$$n_i = M_o \dot{\tilde{\omega}} \tag{5.9}$$

sont les forces et les moments inertiels.

En prenant les dérivées des termes d'énergie potentielle, on obtient

$$\dot{V} = \dot{\tilde{p}}^T K_p \tilde{p} + 2E^T(\tilde{s}, \tilde{v}) \tilde{\omega}^T K_o \tilde{v} \tag{5.10}$$

où

$$f_e = K_p \tilde{p} \tag{5.11}$$

$$n_e = 2E^T(\tilde{s}, \tilde{v}) K_o \tilde{v} \tag{5.12}$$

sont les forces et les moments élastiques. L'équation de propagation des quaternions I.25 a été utilisée pour obtenir n_e.

Finalement les forces et les moments dissipatifs sont

$$f_d = D_p\dot{\tilde{p}} \tag{5.13}$$

$$n_d = D_o\tilde{\omega} \tag{5.14}$$

où D_p et D_o sont des matrices diagonales définies positives caractérisant l'amortissement de l'effecteur.

Finalement, les composantes de translation et de rotation de l'impédance mécanique sont données respectivement par les équations 5.15 et 5.16.

$$M_p\ddot{\tilde{p}} + D_p\dot{\tilde{p}} + K_p\tilde{p} = f \tag{5.15}$$

$$M_o\dot{\tilde{\omega}} + D_o\tilde{\omega} + K'_o\tilde{v} = n \tag{5.16}$$

où

$$K'_o = 2E(\tilde{s},\tilde{v})K_o \tag{5.17}$$

La technique à utiliser pour obtenir une impédance de rigidité non-diagonale est la même que celle présentée ci-haut. Il faut ajouter le terme $2\tilde{s}\tilde{v}K_m\tilde{p}$ à l'énergie potentielle [11], où K_m est la matrice de couplage de la rigidité.

La sélection des matrices de rigidité K_p et K_o influence le comportement de l'effec-

teur dans l'exécution d'une tâche donnée. Analysons donc les termes élastiques d'un point de vue géométrique. La matrice de rigidité K_p peut être décomposée de la façon suivante [9] :

$$K_p = U_p \Gamma_p U_p^T \tag{5.18}$$

où $\Gamma_p = \text{diag}\{\gamma_{p1}, \gamma_{p2}, \gamma_{p3}\}$ est la matrice des valeurs propres et $U_p = [u_{p1}, u_{p2}, u_{p3}]$ est la matrice des vecteurs propres (orthonormaux). Considérons un déplacement de position de longueur λ le long du vecteur propre u_{pi}

$$K_p \tilde{p} = \lambda \gamma_{pi} u_{pi} \tag{5.19}$$

qui est une force élastique le long de l'axe u_{pi}. Ceci implique que la matrice de rigidité de translation, K_p, peut être exprimée en terme de trois paramètres γ_{pi} représentant une rigidité le long des trois axes principaux u_{pi}, permettant ainsi de spécifier la rigidité de translation d'une façon consistante avec la géométrie de la tâche.

D'une façon similaire la relation entre K_o' et K_o peut être analysée en considérant la décomposition suivante de K_o

$$K_o = U_o \Gamma_o U_o^T \tag{5.20}$$

où $\Gamma_o = \text{diag}\{\gamma_{o1}, \gamma_{o3}, \gamma_{o3}\}$ est la matrice des valeurs propres et $U_o = [U_{o1}, U_{o2}, U_{o3}]$ est la matrice des vecteurs propres (orhtonormaux). Considérons un déplacement d'orien-

tation (rotation par un quaternion unitaire) $\{\cos(\theta/2),\ \sin(\theta/2)u_{oi}\}$ autour du vecteur propre u_{oi}

$$K'_o\tilde{\epsilon} = \gamma_{oi}\sin(\theta)u_{oi} \tag{5.21}$$

qui est un couple élastique autour du vecteur u_{oi}. Ceci implique que la matrice de rigidité de rotation, K_o, peut être exprimée à l'aide de trois paramètres, γ_{oi}, représentant une rigidité autour des trois axes principaux u_{oi}, permettant ainsi de spécifier la rigidité de rotation d'une façon consistante avec la géométrie de la tâche.

De façon intuitive nous savons qu'en absence de contact avec l'environnement les trajectoires compliantes, fournies par les impédances de translation et de rotation, convergent vers les trajectoires désirées. Soit les deux fonctions de Lyapunov, pour l'impédance de translation et pour l'impédance de rotation [46] :

$$V_p = T_p + U_p \tag{5.22}$$

$$V_o = T_o + U_o \tag{5.23}$$

Leurs dérivées temporelles en fonction des contributions d'énergie cinétique 5.5 et potentielle 5.6 le long des impédances de translation 5.15 et de rotation 5.16 sont

$$\dot{V}_p = -\dot{\tilde{p}}^T D_p \dot{\tilde{p}} + f^T \dot{\tilde{p}} \tag{5.24}$$

$$\dot{V}_o = -\tilde{\omega}^T D_o \tilde{\omega} + n^T \tilde{\omega} \tag{5.25}$$

Si $f = 0$, alors $\dot{V}_p = 0$ si $\dot{\tilde{p}} = 0$, ce qui implique, par le théorème de La Salle [47] et de l'équation 5.15, que $\tilde{p} = 0$ à l'équilibre. De façon similaire, si $n = 0$, alors $\dot{V}_o = 0$ si $\tilde{\omega} = 0$, ce qui implique, en utilisant les équations 5.16 et 5.17, que $\tilde{v}$ converge vers l'ensemble invariant suivant [10]

$$n = 2\big(\tilde{s}K_o\tilde{v} - S(\tilde{v})K_o\tilde{v}\big) = 0 \tag{5.26}$$

où $S(\cdot)$ est la matrice anti-symétrique du produit vectoriel. Dû à l'opérateur $S(\cdot)$, les deux termes de l'ensemble invariant sont orthogonaux, ce qui donne les deux points d'équilibre suivants

$$\Psi_1 = \{\tilde{s} = 0,\ K_o\tilde{v} = \gamma_o\tilde{v},\ \|\tilde{v}\| = 1\} \tag{5.27}$$

$$\Psi_2 = \{\tilde{s} = \pm 1, \tilde{v} = 0\} \tag{5.28}$$

où γ_o est une valeur propre de la matrice K_o. L'équilibre Ψ_1 est instable : remplaçons $K_o\tilde{v} = \gamma_o\tilde{v}$ dans l'équation 5.23 à l'équilibre pour obtenir

$$V_{o,\infty} = 2\gamma_o\tilde{v}^T\tilde{v} \tag{5.29}$$

Considérons une petite perturbation autour de l'équilibre $\tilde{s} = \epsilon$ tel que $\tilde{v}^T\tilde{v} = 1 - \epsilon^2$, pour obtenir la fonction de Lyapunov perturbée à l'équilibre

$$V_{o,\infty,\epsilon} = 2\gamma_o\big(1 - \epsilon^2\big) < V_{o,\infty} \tag{5.30}$$

On a donc que V_o décroit sans jamais retourner à l'équilibre $V_{o,\infty}$, ce qui implique que l'équilibre Ψ_1 est instable. On conclut que l'équilibre Ψ_2 est stable.

À la section ?? nous introduisons la logique pour faire varier certains paramêtres de l'impédance de translation. Puisque ces changements sont plutôt lent, la démonstration de stabilité de l'impédance de translation demeure valide.

Les impédances de translation 5.15 et de rotation 5.16, ayant comme entrées f et n respectivement, sont intégrées pour donner les trajectoires d'accélérations $\ddot{p}_c$ et $\dot{\omega}_c$, de vitesses $\dot{p}_c$ et ω_c et de positions p_c et v_c. Notons que les équations de propagation des quaternions I.24, I.25 et I.26 ont été utilisées pour obtenir v_c.

Voici les paramètres par défaut des impédances, de translation et de rotation. Notons que l'amortissement est de $\zeta = 1.0$.

Paramètres de l'impédance de translation :

$$M_p = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix} \quad D_p = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 52 \end{bmatrix} \quad K_p = \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 1000 & 0 \\ 0 & 0 & 70 \end{bmatrix} \tag{5.31}$$

Paramètres de l'impédance de rotation :

$$M_o = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix} \quad D_o = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 200 \end{bmatrix} \quad K_o = \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 1000 & 0 \\ 0 & 0 & 1000 \end{bmatrix} \tag{5.32}$$

Les impédances de translation et de rotation doivent êtres jumelées à un contrôleur

de position pour assurer le suivi des trajectoires compliantes. Les deux prochaines sections présentent deux contrôleurs de position différents pouvant êtres utilisés. Rappelons qu'une trajectoire compliante représente la trajectoire permise par les impédances de translation et de rotation pour une trajectoire désirée.

La figure 5.2 illustre une trajectoire cartésienne désirée, alors que les figures 5.3 et 5.4 illustrent des trajectoires compliantes de la trajectoire désirée. Les paramètres de l'impédance de translation de l'équation 5.31 ont été utilisés pour générer la première trajectoire compliante, alors que les paramètres d'impédance de translation de l'équation 5.33 ont produit la deuxième trajectoire compliante.

Nous avons placé deux discontinuités dans la position en z de la trajectoire désirée pour mieux visualiser le concept de trajectoires compliantes. La première discontinuité se produit lorsque la tige (l'effecteur) touche l'environnement alors que la seconde se produit lorsque la tige pénètre dans le trou. On remarque que les trajectoires compliantes sont toujours continues. On constate, en regardant la figure 5.4 que lorsque la rigidité de l'impédance est trop faible, la trajectoire compliante ne peut suivre la trajectoire désirée.

$$M_p = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix} \quad D_p = \begin{bmatrix} 63 & 0 & 0 \\ 0 & 63 & 0 \\ 0 & 0 & 17 \end{bmatrix} \quad K_p = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 7 \end{bmatrix} \tag{5.33}$$

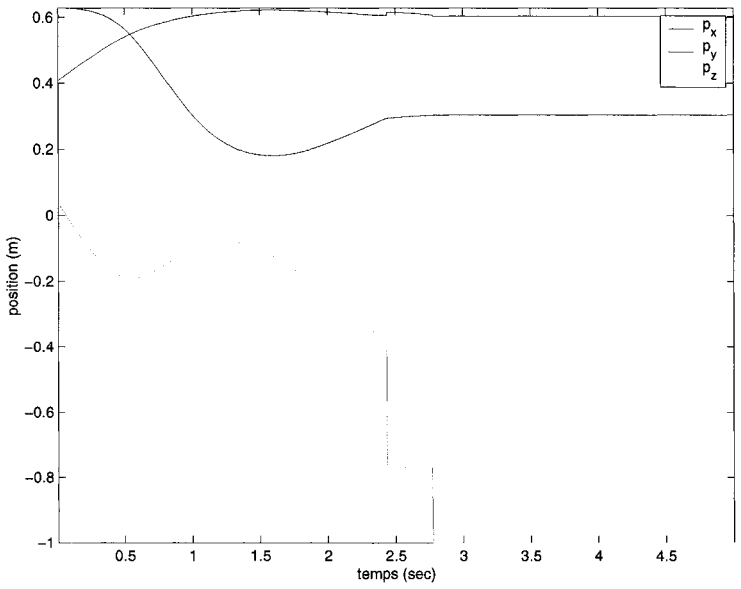


FIG. 5.2 – Positions de translation désirées

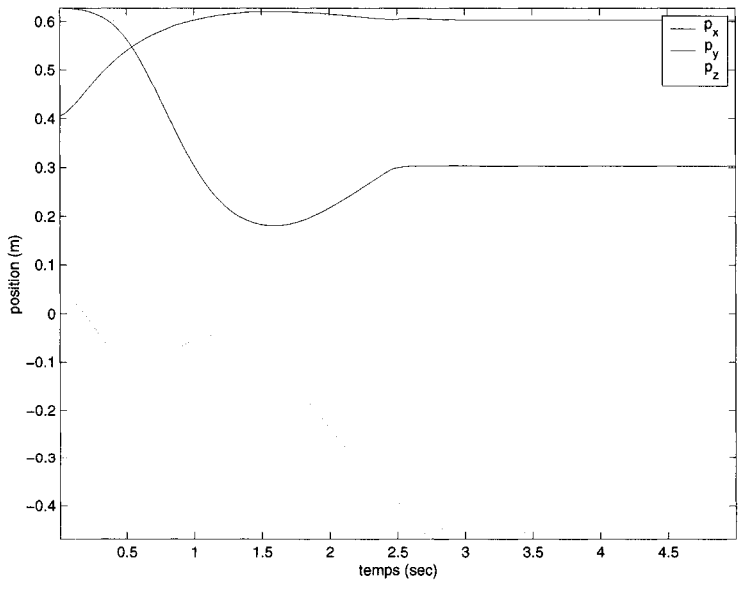


FIG. 5.3 – Positions de translation compliantes obtenues par une impédance ayant les paramètres par défauts, 5.31.

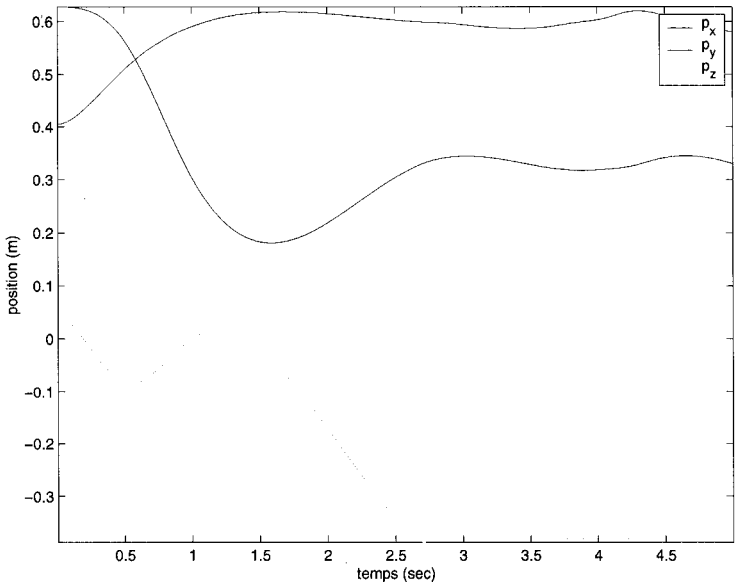


FIG. 5.4 – Positions de translation compliantes obtenues par une impédance ayant les paramètres donnés par l'équation 5.33.

5.2 Contrôle de position selon l'accélération résolue

Nous utilisons un contrôleur établi selon la dynamique inverse pour le suivi des trajectoires compliantes. Rappelons que l'équation de la dynamique d'un robot manipulateur est 3.20. Selon le principe de la dynamique inverse, les couples actifs aux joints (driving torques) peuvent être choisis comme étant

$$\tau = B(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) + J^T(q)f \tag{5.34}$$

En utilisant les équations 3.14 et 5.34 les couples actifs aux joints peuvent s'écrire par [8]

$$\tau = B(q)J^{-1}(q)\big(a - \dot{J}(q,\dot{q})\dot{q}\big) + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) + J^T(q)f \tag{5.35}$$

où a, l'accélération résolue, est la variable de contrôle. Notons qu'une compensation parfaite de la dynamique a été assumée. Cette hypothèse est justifiée puisqu'il est possible d'obtenir le modèle d'un robot manipulateur par des techniques d'identification [1].

L'inverse de la matrice Jacobienne est définie pour un manipulateur non redondant ($n = 6$) se déplaçant dans une région non-singulière de l'espace cartésien. Près d'une singularité, l'inverse selon les moindres carrés amortis (damped least-squares inverse) est utilisée dans cette étude pour une plus grande robustesse [8]. Des explications sur l'inverse selon les moindres carrés amortis est fournie en annexe II.

Le vecteur d'accélération a peut se décomposer en une partie linéaire et une partie angulaire, c-à-d $a = [a_p^T \ a_o^T]^T$. En substituant le contrôleur 5.35 dans l'équation de mouvement 3.20 on obtient

$$\ddot{p}_e = a_p \tag{5.36}$$

$$\dot{\omega}_e = a_o \tag{5.37}$$

où l'indice $_e$ dénote la pose actuelle de l'effecteur. Il faut choisir a_p et a_o de manière à obtenir la pose désirée, identifiée par l'indice $_c$ (trajectoires compliantes). La pose désirée est obtenue par le contrôleur d'impédance.

Analysons l'accélération linéaire résolue a_p en premier lieu. Soit p_c et p_e étant respectivement la position désirée (ou de compliance) et la position actuelle de l'effecteur. Définissons l'erreur de position comme $\Delta p_{ce} = p_c - p_e$. L'accélération désirée peut se

définir par

$$a_p = \ddot{p}_c + k_{vp}\Delta\dot{p}_{ce} + k_{pp}\Delta p_{ce} \tag{5.38}$$

où k_{vp} et k_{pp} sont des gains. En substituant 5.38 dans 5.36 on obtient la dynamique de l'erreur de position en boucle fermée

$$\Delta\ddot{p}_{ce} + k_{vp}\Delta\dot{p}_{ce} + k_{pp}\Delta p_{ce} = 0 \tag{5.39}$$

L'équation 5.39 étant linéaire, elle est exponentiellement stable pour toutes les valeurs de $k_{vp}, k_{pp} > 0$, ce qui assure le suivi de la position desirée.

De son côté l'accélération angulaire résolue peut être obtenue de différentes façons selon la définition choisie pour l'erreur d'orientation de l'effecteur. Soit selon les angles d'Euler, un vecteur et un angle (angle/axis) et les quaternions unitaires. Nous choisissons de préférence l'erreur selon les quaternions unitaires dans cette étude.

Soit la matrice d'erreur de rotation (voir l'équation I.30) exprimant l'orientation désirée (ou de compliance) par rapport à l'orientation actuelle de l'effecteur

$$R_c^e = R_e^T R_c \tag{5.40}$$

L'erreur d'orientation peut être définie par la partie vectorielle du quaternion extrait de R_c^e. La relation entre l'erreur de vitesse angulaire et le quaternion est obtenue par

la règle de propagation des quaternions I.24, I.25 et I.26

$$\dot{s} = -\frac{1}{2} v_{ce}^T \Delta\omega_{ce} \tag{5.41}$$

$$\dot{v} = \frac{1}{2} E(s_{ce}, v_{ce}) \Delta w_{ce} \tag{5.42}$$

L'accélération angulaire résolue peut être donnée par

$$a_o = \dot{\omega}_c + k_{vo}\Delta\omega_{ce} + k_{po}\Delta v_{ce} \tag{5.43}$$

où k_{vo} et k_{po} sont des gains. La dynamique de l'erreur d'orientation est obtenue en substituant l'équation 5.43 dans 5.37

$$\Delta\dot{\omega}_{ce} + k_{vo}\Delta\omega_{ce} + k_{po}v_{ce} = 0 \tag{5.44}$$

Cette dynamique est non linéaire, contrairement à la dynamique de l'erreur de position 5.39. Définissons une fonction candidate de *Lyapunov* pour en analyser la stabilité [8].

$$V = k_{po}\big((s_{ce} - 1)^2 + v_{ce}^T v_{ce}\big) + \frac{1}{2}\Delta\omega_{ce}^T \Delta\omega_{ce} \tag{5.45}$$

La dérivée temporelle de 5.45 le long de trajectoires définies par la dynamique de

l'erreur est

$$\begin{aligned}
\dot{V} &= 2k_{po}\big((s_{ce}-1)\dot{s}_{ce} + v_{ce}^T\dot{v}_{ce}^T\big) + \Delta\omega_{ce}^T\Delta\dot{\omega}_{ce} \qquad (5.46)\\
&= k_{po}\big(-(s_{ce}-1)v_{ce}^T\Delta\omega_{ce} + v_{ce}^T E(s_{ce},v_{ce})\Delta\omega_{ce}\big)\\
&\quad -k_{vo}\Delta\omega_{ce}^T\Delta\omega_{ce} - k_{po}\Delta\omega_{ce}^T v_{ce}\\
&= -k_{vo}\Delta\omega_{ce}^T\Delta\omega_{ce}
\end{aligned}$$

Les équations 5.41 et 5.42 ont été utilisées pour obtenir 5.46.

Il faut faire appel au théorème de *LaSalle* [47] pour l'analyse de stabilité car 5.46 est négative semi-définie. Si par hypothèse $\Delta\dot{\omega}_{ce} = 0$, alors on doit avoir $\Delta\omega_{ce} = 0$. Selon 5.44 ceci implique que $v_{ce} = 0$. De plus selon I.17, on obtient $s_{ce} = \pm 1$. L'ensemble invariant comprend deux points d'équilibre, soit

$$\Psi_1 = \{s_{ce} = -1, v_{ce} = 0, \Delta\omega_{ce} = 0\} \qquad (5.47)$$

$$\Psi_2 = \{s_{ce} = 1, v_{ce} = 0, \Delta\omega_{ce} = 0\} \qquad (5.48)$$

Le point d'équilibre Ψ_1 est instable. Considérons la fonction de *Lyapunov* 5.45, qui est décroissante selon 5.46. À l'équilibre Ψ_1, on a $V_E = 4k_{po}$. Appliquons une légère perturbation $s_{ce} = -1 + \sigma$ (où $\sigma > 0$) autour de l'équilibre. La contrainte suivante doit être respectée pour un quaternion unitaire $s^2 + v^T v = 1$, ce qui dans notre cas donne $v_{ce}^T v_{ce} = 2\sigma - \sigma^2$. La fonction de *Lyapunov* perturbée est

$$V_\sigma = 4k_{po} - 2\sigma k_{po} < V_E \qquad (5.49)$$

V ne retournera pas à l'équilibre puisqu'elle est décroissante, impliquant que Ψ_1 est instable. Le système doit donc converger asymptotiquement vers Ψ_2, impliquant que la poursuite de R_c et ω_c sont atteint.

Paramètres du contrôleur de position selon d'accélération résolue :

$$k_{pp} = k_{po} = 500$$

$$k_{vp} = k_{vo} = 5000$$

5.3 Contrôle de position basé sur la cinématique inverse en boucle fermée

Cette section traite de la seconde approche envisagée pour le contrôle de position. Nous utilisons un contrôleur de couple précalculé accompagné d'une cinématique inverse en boucle fermée.

5.3.1 CLIK

Une solution analytique au problème de la cinématique inverse existe seulement pour des structures non-redondantes et ayant des formes géométriques simples [3]. En poursuivant notre but de fournir de nouveaux outils à la librairie *ROBOOP* [25], nous optons pour une solution qui s'applique à une plus grande classe de robot manipulateur, c.-à-d., la cinématique inverse en boucle fermée (*CLIK* : Closed Loop Inverse Kinematics) [14].

Il existe principalement deux algorithmes *CLIK*, soit celui de premier order [14] et celui de deuxième ordre [7]. Le premier résout la position et la vitesse angulaire des

joints, tandis que le second résout la position, la vitesse et l'accélération angulaire des joints. Par simplicité, nous choisissons d'utiliser le premier algorithme.

Rappelons que la Jacobienne permet de relier les vitesses des joints, $\dot{q}$, aux vitesses cartésiennes de l'effecteur. De façon similaire à la section précédente, l'inverse de la Jacobienne selon l'inverse des moindres carrés est utilisée.

Soit p_d, p, $Q_d = \{s_d, v_d\}$ et $Q = \{s, v\}$ définit comme étant la position désirée, la position actuelle, l'orientation désirée et l'orientation actuelle de l'effecteur. L'erreur d'orientation est obtenue en utilisant l'équation I.33. Comme mentionné à la section I.4, il suffit de vérifier Δv, la partie vectorielle du quaternion représentant l'erreur d'orientation, pour déterminer si Q et Q_d sont allignés. On calcule Q à partir de la matrice de rotation R obtenue par la cinématique directe I.20. L'erreur de position est donnée par $\delta p = p_d - p$.

Selon l'algorithme on définie la vitesse angulaire des joints comme étant [14]

$$\dot{q} = J^{-1}(q) \begin{bmatrix} \dot{p}_d + C_p \delta p \\ w_d + C_o \Delta v \end{bmatrix} \tag{5.50}$$

où C_p et C_o sont des matrices diagonales définies positives et w_d est la vitesse angulaire désirée de l'effecteur. Les vitesses angulaires des joints sont ensuite intégrées pour obtenir la position angulaire des joints. Les calculs d'erreurs de position et d'orientation sont utilisés pour contrer la dérive numérique due à l'intégration.

La figure 5.5 illustre le schéma bloque de l'algorithme *CLIK*.

Analysons maintenant la stabilité de l'algorithme. On obtient les équations différentielles

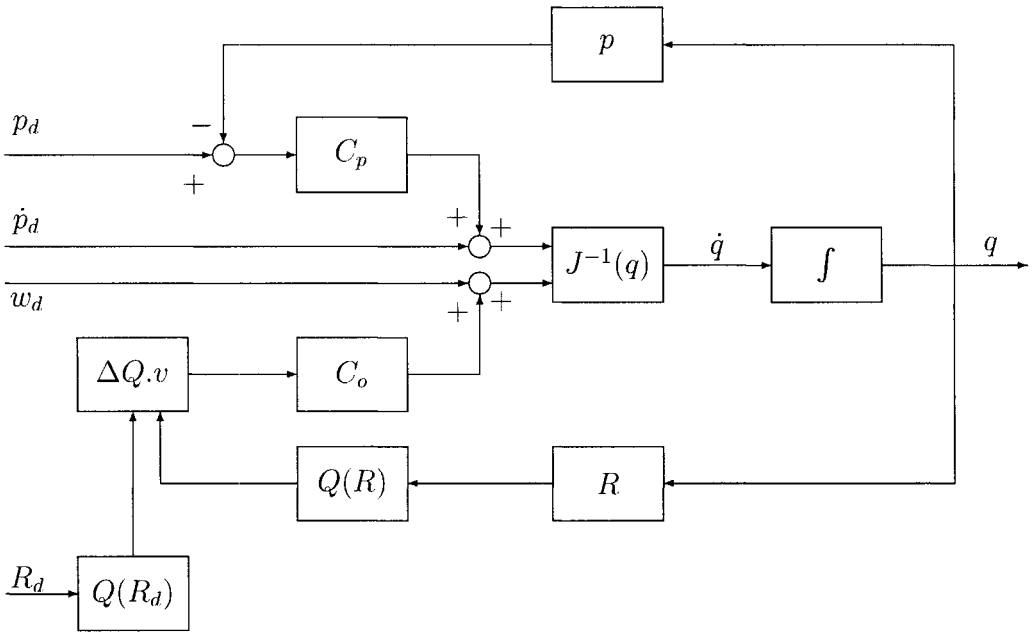


FIG. 5.5 Schéma de l'algorithme CLIK.

de l'erreur en substituant 5.50 dans 3.9.

$$\delta\dot{p} + C_p\delta p = 0 \tag{5.51}$$

$$w_d - w + C_o\Delta v = 0 \tag{5.52}$$

L'équation 5.51 est exponentiellement stable puisqu'elle est linéaire. Par contre 5.52 est non-linéaire puisqu'elle contient l'erreur de la vitesse angulaire de l'effecteur au lieu de la dérivée de l'erreur d'orientation. Considérons la fonction de Lyapunov

suivante pour en analyser la stabilité

$$V = (s_d - s)^2 + (v_d - v)^T (v_d - v) \tag{5.53}$$

$$\dot{V} = -\Delta v C_o \Delta v \tag{5.54}$$

La dérivée de 5.53 selon la dynamique de l'erreur d'orientation 5.52 est définie négative, impliquant que l'erreur d'orientation tend vers 0. Les équations de propagation des quaternions I.24 I.25 ont été utilisées pour obtenir 5.54.

Les paramètres de la méthode *CLIK* utilisés sont

$$K_p = \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 1000 & 0 \\ 0 & 0 & 1000 \end{bmatrix} \quad K_o = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix}$$

L'algorithme *CLIK* est implanté par la classe *CLIK* de *ROBOOP*.

5.3.2 Couple précalculé

Le contrôleur de couple précalculé est donné par l'équation suivante :

$$\tau = B(q)\big(\ddot{q}_d + H_d(\dot{q}_d - \dot{q}) + H_p(q_d - q)\big) + C(q, \dot{q})\dot{q} + D\dot{q} + g(q) - J^T(q)f \tag{5.55}$$

où H_d et H_p sont des matrices diagonales définies positives. Il est à noter que $\ddot{q}_d$ est toujours nul puisque nous employons l'algorithme de cinématique inverse de premier

ordre.

Les paramètres utilisés du contrôleur sont

$$K_p = \begin{bmatrix} 1000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix} \quad K_d = \begin{bmatrix} 50 & 0 & 0 & 0 & 0 & 0 \\ 0 & 350 & 0 & 0 & 0 & 0 \\ 0 & 0 & 350 & 0 & 0 & 0 \\ 0 & 0 & 0 & 150 & 0 & 0 \\ 0 & 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}$$

5.4 Mise en oeuvre

Nous avons déjà mentionné que nous utilisons l'inverse de la Jacobienne selon la technique des moindres carrés amortis pour éviter une région singulière. Par contre, s'il est difficile d'utiliser cette technique dans une application donnée, on peut simplement calculer le déterminant (où $det(JJ^T$ si le robot n'a pas 6 joints) de la Jacobienne. Lorsque le déterminant est plus petit qu'une certaine valeur, on arrête le contrôle du robot dans l'espace cartésien.

La figure 5.6 illustre l'oscillation de la position en z de l'effecteur, pour différentes valeurs du facteur d'amortissement ζ de l'impédance de translation, lorsque ce dernier est soumis à une force perturbatrice en z de très courte durée (impulsion). Les résultats représentent bien la réponse impulsionnelle d'un système de deuxième ordre. La figure 5.7 illustre la position de l'effecteur soumis à la même perturbation lorsque le facteur d'amortissement est de 0.1. Un point de singularité (les joints 2, 3 et 4 sont colinéaires)

est atteint lors de l'oscillation. La stabilité, lors du passage par le point de singularité, est assuré grâce à l'inverse de la Jacobienne selon les moindres carrés. Le déplacement en x et y est dû à cet inverse. Le robot était dans une position similaire à celle illustrée à la figure 3.2 avant l'atteinte de la singularité, alors qu'il se trouve dans une position similaire à celle illustrée à la figure 5.8. Rappelons que l'inverse selon les moindres carrés amortis est détaillé à l'annexe II.

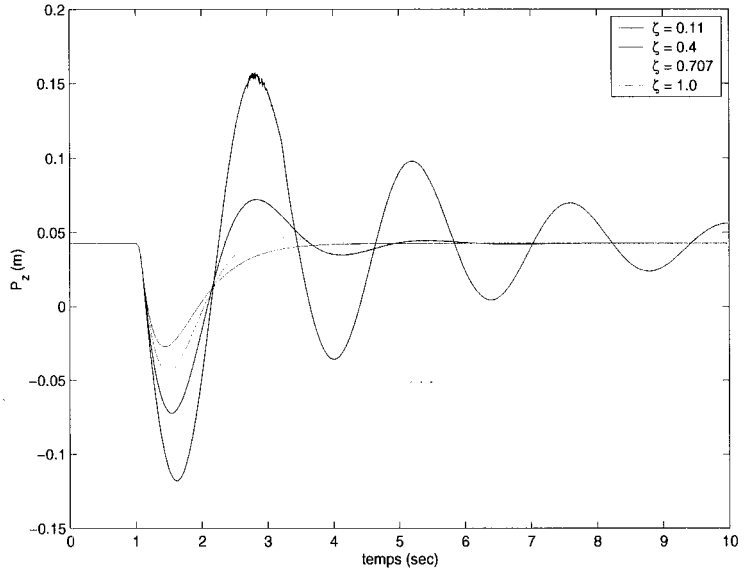


FIG. 5.6 – Position en z de l'effecteur soumis à une perturbation pour différentes valeurs d'amortissement.

Le contrôleur d'impédance est implanté dans les classes *Impedance* et *Resolved_acc* de *ROBOOP*.

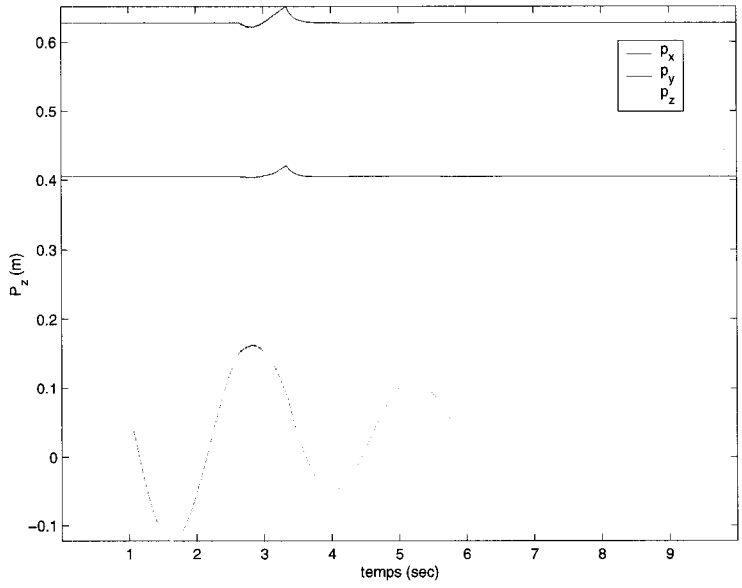


FIG. 5.7 – Position de l'effecteur soumis à une perturbation pour une amortissement de $\zeta = 0.1$.

5.5 Observateur des vitesses angulaires

Nous avons mentionné, dans l'introduction, que les joints du robot simulé ne portaient pas de capteur de vitesses angulaires. Dans cette section, nous présentons l'observateur des vitesses angulaires des joints élaboré par Nicosia et Tomei [41]. À la section 5.6, nous montrons que l'agencement de l'observateur et du contrôleur d'accélération résolue est stable, alors qu'à la section 5.6 nous montrons que l'agencement de l'observateur et du contrôleur de couples précalculés est stable. Nous croyons que ces démonstrations sont nouvelles.

Assumons que l'équation de mouvement détermine la position angulaire des joints. Dans le but de simplifier, nous négligeons les effets de la friction de Coulomb car cette dernière est fonction du signe de $\dot{q}$. L'équation de mouvement utilisée dans

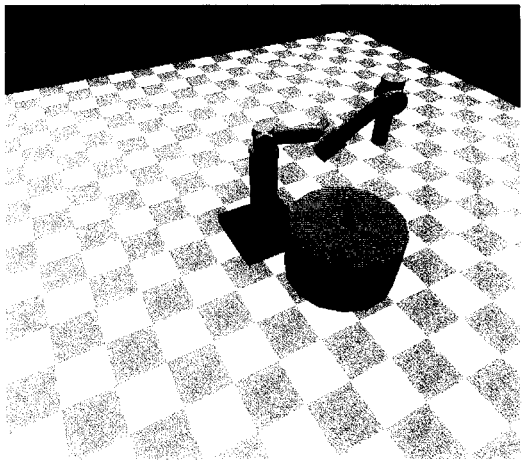


FIG. 5.8 – Position du robot après avoir traversé une singularité.

le développement de l'observateur est 3.20. Par contre, cette fois-ci la matrice D représente uniquement les effets visqueux.

Définissons l'erreur d'observation par

$$\tilde{y} = q - \hat{y} \tag{5.56}$$

et l'observateur par

$$\dot{\hat{x}}_1 = \hat{x}_2 + k_d\tilde{y} \tag{5.57}$$

$$\dot{\hat{x}}_2 = B^{-1}(q)\big[-C(q,\dot{\hat{x}}_1)\dot{\hat{x}}_1 + \tau - J^T(q)f + k_p\tilde{y} - g(q) - D\dot{\hat{x}}_1\big] \tag{5.58}$$

$$\hat{q} = \hat{x}_1 \tag{5.59}$$

où k_d est un scalaire positif constant et k_p est une matrice symétrique définie positive constante.

Nicosia et Tomei [41] ont montré que cet observateur est stable. Posons $x^T = [\tilde{y}^T, \dot{\tilde{y}}^T]$ et $L(q)$ une matrice diagonale $[k_p, B(q)]$. Supposons que $\|\dot{q}(t)\| \leq k_q$ pour tout $t \geq 0$. De plus, si nous avons

$$k_d > \frac{k_c k_q - D_m}{B_m} \tag{5.60}$$

où B_m est la plus petite valeur propre de la matrice d'inertie B, alors le point d'équilibre, $x = 0$, est asymptotiquement stable et une région d'attraction est donnée par

$$S = \left\{ x \in \mathbf{R}^{2n} : \|x\| < \sqrt{\frac{L_m}{L_M}} \left(\frac{k_d B_m + D_m}{k_c} - k_q \right) \right\} \tag{5.61}$$

Les développements subséquents de cette section démontrent la stabilité de l'observateur [41]. Due à la complexité du système, la stabilité démontrée est locale. Le résultat est sans doute un peu conservateur.

En combinant la dérivée temporelle de 5.57 et les équations 5.58 5.59 on obtient

$$B(q)\ddot{\hat{q}} + C(q, \dot{\hat{q}})\dot{\hat{q}} + D\dot{\hat{q}} + g(q) = \tau - J^T(q)f + k_p\tilde{q} + k_d B(q)\dot{\tilde{q}} \tag{5.62}$$

La dynamique de l'erreur est obtenue en soustrayant 5.62 de 3.20

$$B(q)\ddot{\tilde{q}} + C(q, \dot{q})\dot{q} - C(q, \dot{\hat{q}})\dot{\hat{q}} + D\dot{\tilde{q}} = -k_p\tilde{q} - k_d B(q)\dot{\tilde{q}} \tag{5.63}$$

En utilisant la propriété p2 de la section 3.2 on obtient l'égalité suivante

$$\begin{aligned} C(q,\dot{q})\dot{q} - C(q,\dot{\hat{q}})\dot{\hat{q}} &= C(q,\dot{q})\dot{q} - C(q,\dot{q})\dot{\hat{q}} + C(q,\dot{q})\dot{\hat{q}} - C(q,\dot{\hat{q}})\dot{\hat{q}} \\ &= C(q,\dot{q})\dot{\tilde{q}} + C(q,\dot{\tilde{q}})\dot{\tilde{q}} \end{aligned} \tag{5.64}$$

et en substituant 5.64 dans la dynamique de l'erreur 5.63 on obtient

$$B(q)\ddot{\tilde{q}} + C(q,\dot{q})\dot{\tilde{q}} + D\dot{\tilde{q}} = -k_p\tilde{q} - k_d B(q)\dot{\tilde{q}} - C(q,\dot{\tilde{q}})\dot{\tilde{q}} \tag{5.65}$$

Considérons maintenant la fonction de Lyapunov

$$V(x,t) = \frac{1}{2}x^T L(q)x \tag{5.66}$$

La dérivée temporelle de 5.66 le long de 5.65 est

$$\begin{aligned} \dot{V}(x,t) &= x^T L(q)\dot{x} + \frac{1}{2}x^T \dot{L}(q)x \\ &= \begin{bmatrix} \tilde{q}^T & \dot{\tilde{q}}^T \end{bmatrix} \begin{bmatrix} k_p & 0 \\ 0 & B(q) \end{bmatrix} \begin{bmatrix} \dot{\tilde{q}} \\ \ddot{\tilde{q}} \end{bmatrix} + \\ &\quad \frac{1}{2}\begin{bmatrix} \tilde{q}^T\dot{\tilde{q}}^T \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & \dot{B}(q) \end{bmatrix} \begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{bmatrix} \\ &= \dot{\tilde{q}}^T[-C(q,\dot{q})\dot{\tilde{q}} - C(q,\dot{\tilde{q}})\dot{\tilde{q}} - D\dot{\tilde{q}} - k_p\tilde{q} - k_d B(q)\dot{\tilde{q}}] + \\ &\quad \frac{1}{2}\dot{\tilde{q}}^T\dot{B}(q)\dot{\tilde{q}} + \dot{\tilde{q}}^T k_p\tilde{q} \end{aligned} \tag{5.67}$$

Eu utilisant la propriété p1 de la section 3.2 l'équation 5.67 devient

$$\dot{V}(x,t) = -\dot{\tilde{q}}^T(k_dB + D)\dot{\tilde{q}} - \dot{\tilde{q}}^T C(q,\dot{\hat{q}})\dot{\tilde{q}} \tag{5.68}$$

En utilisant la propriété p2 de la section 3.2 on peut réécrire le dernier terme de l'équation précédente

$$\begin{aligned} C(q,\dot{\hat{q}})\dot{\tilde{q}} &= C(q,\dot{\tilde{q}})\dot{\hat{q}} \\ &= C(q,\dot{\tilde{q}})(\dot{q} - \dot{\tilde{q}}) \\ &= -C(q,\dot{\tilde{q}})\dot{\tilde{q}} + C(q,\dot{\tilde{q}})\dot{q} \\ &= -C(q,\dot{\tilde{q}})\dot{\tilde{q}} + C(q,\dot{q})\dot{\tilde{q}} \end{aligned} \tag{5.69}$$

Par les propriétés p2 et p3 de la section 3.2 on obtient

$$|\dot{\tilde{q}}^T C(q,\dot{\hat{q}})\dot{\tilde{q}}| = |-\dot{\tilde{q}}^T C(q,\dot{\tilde{q}})\dot{\tilde{q}} + \dot{\tilde{q}}C(q,\dot{q})\dot{\tilde{q}}| \leq \|\dot{\tilde{q}}\|^2 k_c(\|\dot{\tilde{q}}\| + k_q) \tag{5.70}$$

où $\|x\|$ est la norme du vecteur x. À partir de cette étape Nicosia et Tomei [41] ne tiennent plus compte de la contribution de l'amortissement visqueux. Nous n'avons pas fait cette simplification. L'équation 5.70 permet de simplifier la fonction de Lyapunov pour donner

$$\dot{V}(x,t) \leq -\|\dot{\tilde{q}}\|^2\left[k_dB_m + D_m - k_c(\|\dot{\tilde{q}}\| + k_q)\right] \tag{5.71}$$

Cette fonction de Lyapunov est définie négative si

$$\|\dot{\tilde{q}}\| < \frac{k_d B_m + D_m}{k_c} - k_q \tag{5.72}$$

alors

$$\dot{v} \leq -\beta \|\dot{\tilde{q}}\|^2 \tag{5.73}$$

où β est une constante positive. Par hypothèse (voir la définition de k_d 5.60) le côté droit de 5.72 est positif. De plus,

$$\frac{1}{2} L_m \|x\|^2 \leq V(x,t) \leq \frac{1}{2} L_M \|x\|^2 \tag{5.74}$$

De l'équation 5.74 on voit que $V(x,t)$ est une fonction décroissante définie positive. Comme $\dot{V}(x,t)$ est une fonction négative semi-définie pour tout x satisfaisant 5.72, alors on peut conclure que le point d'équilibre $x = 0$ est uniformément stable [47]. La région d'attraction est obtenue par les équations 5.72 à 5.74 [41]

$$\|x(0)\| < \sqrt{\frac{L_m}{L_M}} \Big(\frac{k_d B_m + D_m}{k_c} - k_q \Big) \tag{5.75}$$

La convergence de l'observateur est assurée seulement si les vitesses angulaires des joints sont bornées et que les conditions initiales appartiennent à une région d'attraction. La région d'attaction peut être agrandie en augmentant le gain k_d. De plus, puisque la position angulaire des joints est connue, seul $\dot{\tilde{y}}(0)$ (estimé initial des vitesses

angulaires) doit satisfaire la région d'attraction.

5.6 Stabilité de l'observateur et du contrôleur d'accélération résolue

Les développements subséquents sont applicables lorsque la matrice Jacobienne n'est pas singulière.

En combinant l'équation du contrôleur 5.35 à celle de la dynamique du robot 3.20 et en remplaçant $\dot{q}$ par $\dot{\hat{q}}$ (estimé de la vitesse des joints) nous obtenons

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) = B(q)J^{-1}(q)\big(a - \dot{J}(q,\dot{\hat{q}})\dot{\hat{q}}\big) \tag{5.76}$$

$$+C(q,\dot{\hat{q}})\dot{\hat{q}} + D\dot{\hat{q}} + g(q) \tag{5.77}$$

En utilisant 3.14 remplaçons $\ddot{q}$ dans l'expression précédente par

$$\ddot{q} = J^{-1}(q)\big[a_e - \dot{J}(q,\dot{q})\dot{q}\big] \tag{5.78}$$

pour obtenir à l'aide de la proprité p5 de la section 3.2

$$B(q)J^{-1}(q)\big[\tilde{a} - \dot{J}(q,\dot{q})\dot{\tilde{q}} - \dot{J}(q,\dot{\hat{q}})\dot{\tilde{q}}\big] = -C(q,\dot{q})\dot{q} - C(q,\dot{\tilde{q}})\dot{\hat{q}} - D\dot{\tilde{q}} \tag{5.79}$$

où a_e est l'accélération de l'effecteur et $\tilde{a} = a_e - a$. Le terme $\tilde{a}$ représente l'erreur de poursuite du contrôleur.

Posons

$$u(t) = J(q)B^{-1}(q)\big[-C(q,\dot{\tilde{q}}) - D\big]\dot{\tilde{q}} + \dot{J}(q,\dot{q})\dot{\tilde{q}} + \dot{J}(q,\dot{\tilde{q}})\dot{\tilde{q}} \tag{5.80}$$

L'erreur d'accélération $\tilde{a}$ se décompose en une composante de translation et une composante de rotation.

$$\tilde{a} = u(t) \tag{5.81}$$

$$\begin{bmatrix} \tilde{a}_p \\ \tilde{a}_o \end{bmatrix} = \begin{bmatrix} u_p(t) \\ u_o(t) \end{bmatrix} \tag{5.82}$$

$$\ddot{\tilde{p}} + k_{vp}\dot{\tilde{p}} + k_{pp}\tilde{p} = u_p \tag{5.83}$$

$$\dot{\tilde{\omega}} + k_{vo}\tilde{\omega} + k_{po}\tilde{v} = u_o \tag{5.84}$$

où $\tilde{p} = p_d - p_e$ et $\tilde{\omega} = \omega_d - \omega_e$.

$$V = \frac{1}{2}x^T M x + k_{po}\big((\tilde{s}-1)^2 + \tilde{v}^T\tilde{v}\big) + \frac{1}{2}\tilde{\omega}^T\tilde{\omega} \tag{5.85}$$

où $x^T = [\tilde{p}^T,\ \dot{\tilde{p}}^T,\ \tilde{q}^T,\ \dot{\tilde{q}}^T]$ et

$$M = \begin{bmatrix} M_1 & 0 & 0 & 0 \\ 0 & M_2 & 0 & 0 \\ 0 & 0 & k_p & 0 \\ 0 & 0 & 0 & B(q) \end{bmatrix}$$

La dérivée temporelle de la fonction de Lyapunov le long de la dynamique de l'erreur d'observation 5.63 et le long des dynamiques des erreurs de poursuite de translation 5.83 et de rotation 5.84 du contrôleur est

$$\begin{aligned} \dot{V} &= \tilde{p}^T M_1 \dot{\tilde{p}} + \dot{\tilde{p}}^T M_2 \ddot{\tilde{p}} + \tilde{q}^T k_p \dot{\tilde{q}}^T + \dot{\tilde{q}}^T B(q) \ddot{\tilde{q}} + \frac{1}{2} \dot{\tilde{q}}^T \dot{B}(q) \dot{\tilde{q}} \\ &\quad + 2k_{po}\big((\tilde{s}-1)\dot{\tilde{s}} + \tilde{v}^T \dot{\tilde{v}}\big) + \tilde{\omega}^T \dot{\tilde{\omega}} \end{aligned} \tag{5.86}$$

En substituant les termes $\ddot{\tilde{p}}$ et $\dot{\tilde{\omega}}$ par les équations d'erreur de la dynamique 5.83 et 5.84, et en remplaçant les termes $\dot{\tilde{s}}$ et $\dot{\tilde{v}}$ par les équations de propagation des quaternions I.24, I.25 et I.26 nous obtenons

$$\begin{aligned} \dot{V} &= \tilde{p}^T M_1 \dot{\tilde{p}} + \dot{\tilde{p}}^T M_2 \big[u_p - k_{vp}\dot{\tilde{p}} - k_{pp}\tilde{p}\big] + \tilde{q}^T k_p \dot{\tilde{q}} + \frac{1}{2} \dot{\tilde{q}}^T \dot{B}(q) \dot{\tilde{q}} + \\ &\quad \dot{\tilde{q}}^T \big[- k_p \tilde{q} - k_d B(q)\dot{\tilde{q}} - C(q,\dot{q})\dot{\tilde{q}} - D\dot{\tilde{q}} - C(q,\dot{q})\dot{\tilde{q}}\big] + \\ &\quad k_{po}\big[- (\tilde{s}-1)\tilde{v}^T\tilde{\omega} + \tilde{v}^T\big(\tilde{s}I + S(\tilde{v})\big)\tilde{\omega}\big] + \tilde{\omega}^T\big[u_o - k_{vo}\tilde{\omega} - k_{po}\tilde{v}\big] \end{aligned} \tag{5.87}$$

Eu utilisant la propriété p1 de la section 3.2 et le fait que $\tilde{v}^T S(\tilde{v}) = 0$ nous obtenons

$$\begin{aligned} \dot{V} &= \dot{\tilde{p}}^T \big(M_1 - k_{pp} M_2\big)\tilde{p} + \dot{\tilde{p}}^T M_2 u_p - \dot{\tilde{p}}^T k_{vp} M_2 \dot{\tilde{p}} - \\ &\quad \dot{\tilde{q}}^T \big[k_d B(q) + C(q,\dot{\hat{q}}) + D\big]\dot{\tilde{q}} - k_{vo}\tilde{\omega}^T\tilde{\omega} + \tilde{\omega}^T u_o \end{aligned} \tag{5.88}$$

En choisissant $M_1 = k_{pp} M_2$ l'équation précédante devient

$$\begin{aligned} \dot{V} &= -\Big[\dot{\tilde{p}}^T k_{vp} M_2 \dot{\tilde{p}} + \dot{\tilde{q}}^T \big[k_d B(q) + D\big]\dot{\tilde{q}} + k_{vo}\tilde{\omega}^T\tilde{\omega}\Big] \\ &\quad + \dot{\tilde{q}}^T C(q,\dot{\hat{q}})\dot{\tilde{q}} + \tilde{\omega}^T u_o + \dot{\tilde{p}}^T M_2 u_p \end{aligned} \tag{5.89}$$

En utilisant l'inégalité du triangle pour l'équation précédente nous obtenons

$$\begin{aligned} \dot{V} &\leq -\Big[\|\dot{\tilde{p}}\|^2 k_{vp} M_{2_m} + \|\dot{\tilde{q}}\|^2 \big(k_d B_m + D_m\big) + \|\tilde{\omega}\|^2 k_{vo}\Big] \\ &\quad + \|\dot{\tilde{q}}\|^2 k_c \|\dot{\hat{q}}\| + \|\tilde{\omega}\| u_k + \|\dot{\tilde{p}}\| M_{2_M} u_k \end{aligned} \tag{5.90}$$

où u_k est la plus grande norme de $u(t)$. À l'aide des propriétés p4 et p6 de la section 3.2 et de l'hypothèse $\|\dot{q}\| \leq k_q$

$$u_k \leq k_p k_2 k_c \|\dot{\tilde{q}}\|^2 + D_M \|\dot{\tilde{q}}\| + k_j k_q \|\dot{\tilde{q}}\| + k_j \|\dot{\hat{q}}\| \|\dot{\tilde{q}}\| \tag{5.91}$$

En regroupant certains termes nous obtenons

$$\begin{aligned}\dot{V} \leq\ & \|\dot{\tilde{p}}\| M_{2_M}\Big[\big(D_M + k_j k_q + k_j(\|\dot{q}\|+k_q)\big)\|\dot{\tilde{q}}\| - \|\dot{\tilde{p}}\| k_{vp} M_{2_m}\Big] \\ & + \|\tilde{\omega}\|\Big[\big(D_M + k_j k_q + k_j(\|\dot{q}\|+k_q)\big)\|\dot{\tilde{q}}\| - \|\tilde{\omega}\| k_{vo}\Big] \\ & + \|\dot{\tilde{q}}\|^2\Big[- k_d B_m - D_m + \|\dot{\hat{q}}\| k_c + k_p k_2 k_c \|\tilde{\omega}\| + k_p k_2 k_c \|\dot{\tilde{p}}\| M_{2_M}\Big]\end{aligned} \tag{5.92}$$

où $\|\dot{\hat{q}}\| \leq \|\dot{\tilde{q}}\| + \|\dot{q}\| \leq \|\dot{q}\| + k_q$

Le système est stable pour une valeur de k_d donnée par l'équation 5.95 et pour kvp et k_{vo} assez grand.

$$k_{vp}\|\dot{\tilde{p}}\| > \frac{\big(D_M + k_j k_q + k_j(\|\dot{\tilde{q}}\|+k_q)\big)\|\dot{\tilde{q}}\|}{M_{2_m}} \tag{5.93}$$

$$k_{vo}\|\tilde{\omega}\| > \big(D_M + k_j k_q + k_j(\|\dot{\tilde{q}}\|+k_d)\big)\|\dot{\tilde{q}}\| \tag{5.94}$$

$$k_d > \frac{(\|\dot{\tilde{q}}\|+k_q)k_c + k_p k_2 k_c\|\tilde{\omega}\| + k_p k_2 k_c\|\dot{\tilde{p}}\| M_{2_M} - D_m}{B_m} \tag{5.95}$$

En terme de région de stabilité nous obtenons

$$\|\tilde{\omega}\| + \|\dot{\tilde{p}}\| M_{2_M} < \frac{B_m k_d - (\|\dot{\tilde{q}}\|+k_q)k_c + D_m}{k_p k_2 k_c} \tag{5.96}$$

5.7 Stabilité de l'observateur et du contrôleur par couple précalculé

De façon similaire à la section précédente, nous démontrons la stabilité du tandem observateur et contrôleur par couple précalculé. Cette démonstration semble ne jamais

avoir été faite.

En combinant l'équation du contrôleur 5.55 à celle de la dynamique du robot 3.20 et en remplaçant le vecteur de vitesse angulaire $\dot{q}$ par son estimé, $\dot{\hat{y}}$, nous obtenons

$$\begin{aligned} B(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) &= B(q)\big(\ddot{q}_d + H_d(\dot{q}_d - \dot{\hat{y}}) + H_p(q_d - q)\big) \\ &\quad + C(q,\dot{\hat{y}})\dot{\hat{y}} + D\dot{\hat{y}} + g(q) \qquad (5.97) \\ C(q,\dot{q})\dot{q} + D\dot{\tilde{q}} - C(q,\dot{\hat{y}})\dot{\hat{y}} &= B(q)\Big(\ddot{\tilde{q}} + H_d\dot{\tilde{q}} + H_d\dot{\tilde{y}} + H_p\tilde{p}\Big) \qquad (5.98) \end{aligned}$$

où $\tilde{y} = q - \hat{y}$ et $\tilde{q} = q_d - q$.

Soit la fonction de Lyapunov suivante

$$V = x^T M x \qquad (5.99)$$

où $x^T = [\tilde{q}^T,\ \dot{\tilde{q}}^T,\ \tilde{y}^T,\ \dot{\tilde{y}}^T]$ et

$$M = \begin{bmatrix} H_p & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & k_p & 0 \\ 0 & 0 & 0 & B(q) \end{bmatrix}$$

La dérivée temporelle de la fonction de Lyapunov selon la dynamique d'erreur de

l'observateur 5.65 et la dynamique d'erreur du contrôleur 5.98 est

$$\begin{aligned}
\dot{V} &= x^T M\dot{x} + \frac{1}{2}x^T\dot{M}x \\
&= \tilde{q}^T H_p\dot{\tilde{q}} + \dot{\tilde{q}}^T\ddot{\tilde{q}} + \tilde{y}^T k_p\dot{\tilde{y}} + \dot{\tilde{y}}^T B(q)\ddot{\tilde{y}} + \frac{1}{2}\dot{\tilde{y}}^T\dot{B}(q)\dot{\tilde{y}} \\
&= \tilde{q}^T H_p\dot{\tilde{q}} + \tilde{y}^T k_p\dot{\tilde{y}}^T + \frac{1}{2}\dot{\tilde{y}}^T\dot{B}\dot{\tilde{y}} \qquad (5.100) \\
&\quad +\dot{\tilde{q}}^T\Big[B^{-1}\big(C(q,\dot{q})\dot{q} + D\dot{\tilde{y}} - C(q,\dot{\hat{y}})\dot{\hat{y}}\big) - H_p\tilde{q} - H_d\dot{\tilde{q}} - H_d\dot{\tilde{y}}\Big] \\
&\quad +\dot{\tilde{y}}^T\Big[- C(q,\dot{q})\dot{\tilde{y}} - D\dot{\tilde{y}} - k_p\tilde{y} - k_d B(q)\dot{\tilde{y}} - C(q,\dot{\hat{y}})\dot{\tilde{y}}\Big]
\end{aligned}$$

En utilisant la propriété p1 de la section 3.2 et l'équation 5.64, $\dot{V}$ devient

$$\begin{aligned}
\dot{V} &= \dot{\tilde{q}}^T B^{-1}(q)\Big[C(q,\dot{q})\dot{\tilde{y}} + D\dot{\tilde{y}} + C(q,\dot{\hat{y}})\dot{\tilde{y}}\Big] - \dot{\tilde{q}}^T H_d\dot{\tilde{q}} - \dot{\tilde{q}}^T H_d\dot{\tilde{y}} + \\
&\quad \dot{\tilde{y}}^T\Big[- D\dot{\tilde{y}} - k_d B(q) - C(q,\dot{\tilde{y}})\Big]\dot{\tilde{y}}
\end{aligned}$$

Les propriétés p2 et p3 de la section 3.2 et l'hypothèse $\|\dot{q}(t)\| \leq k_q$ impliquent que

$$|C(q,\dot{\hat{y}})\dot{\tilde{y}}| = |-C(q,\dot{\hat{y}})\dot{\tilde{y}} + C(q,\dot{q})\dot{\tilde{y}}| \leq \|\dot{\tilde{y}}\|k_c(\|\dot{\tilde{y}}\| + k_q) \qquad (5.101)$$

En utilisant l'équation précédente et la propriété p4 de la section 3.2, nous obtenons l'inégalité du triangle suivante :

$$\begin{aligned}
\dot{V} &\leq -\|\dot{\tilde{q}}\|^2 H_{d_m} - \|\dot{\tilde{y}}\|^2\Big[D_m + k_d B_m\Big] + \|\dot{\tilde{y}}\|^2 k_c(\|\dot{\tilde{y}}\| + k_q) \qquad (5.102) \\
&\quad +\|\dot{\tilde{y}}\|\|\dot{\tilde{q}}\|\Big[H_{d_M} + k_i\big(D_M + k_c k_q + k_c(\|\dot{\tilde{y}}\| + k_q)\big)\Big]
\end{aligned}$$

Rappelons que X_M et X_m sont respectivement la plus grande et la plus petite valeur propre de la matrice X. Sous forme matricielle nous obtenons

$$\dot{V} \leq -\begin{bmatrix} \|\dot{\tilde{q}}\| \\ \|\dot{\tilde{y}}\| \end{bmatrix}^T \begin{bmatrix} H_{d_m} & A - \frac{1}{2}k_c\|\dot{\tilde{y}}\| \\ A - \frac{1}{2}k_c\|\dot{\tilde{y}}\| & E - k_c\|\dot{\tilde{y}}\| \end{bmatrix} \begin{bmatrix} \|\dot{\tilde{q}}\| \\ \|\dot{\tilde{y}}\| \end{bmatrix} \tag{5.103}$$

où $A = -\frac{1}{2}\Big(H_{d_M} + k_i\big[2k_ck_q + D_M\big]\Big)$ et $E = \big(D_m + k_dB_m - k_ck_q\big)$.

Trouvons les conditions qui rendent la matrice précédente définie positive, c.-à-d. un déterminant positif.

$$-\frac{k_c^2}{4}\|\dot{\tilde{y}}\|^2 - \big(H_{d_m}k_c - Ak_c\big)\|\dot{\tilde{y}}\| + H_{d_m}E - A^2 > 0 \tag{5.104}$$

$$\|\dot{\tilde{y}}\| \quad < \quad \frac{2H_{d_m}(\sqrt{G} - F)}{k_c} \tag{5.105}$$

où

$$\begin{aligned} F &= \Big[1 + \frac{H_{d_M} + 2k_ik_ck_q + k_iD_M}{H_{d_m}}\Big] \\ G &= \Big[\frac{1 + H_{d_M} + 2k_ik_ck_q + k_iD_M + D_m + k_dB_m - k_ck_q}{H_{d_m}}\Big] \end{aligned} \tag{5.106}$$

Le terme $\sqrt{G} - F$ doit être positif pour qu'il y ait stabilité. Par inspection, nous voyons que cela est possible pour une valeur assez élevée du gain k_d.

CHAPITRE 6

STRATÉGIES

À la section 6.1, nous verrons comment l'utilisation de la logique floue peut faire varier la rigidité de l'impédance, permettant de mieux adapter le contrôleur aux différentes phases de travail. La section 6.2 présente les équations pour la génération de trajectoires de translations par splines cubiques paramétriques et pour la génération de trajectoires de rotations par splines de quaternions. Le robot simulé ne comportant pas de système de vision, une technique de recherche aveugle du trou lors de l'insertion a été envisagée. La section 6.3 décrit une méthode de recherche basée sur un tracé en spirale. On y présente également une méthode permettant l'insertion, même si la surface du trou n'est pas parallèle au plan $x-y$ du repère de base. Finalement la section 6.4 est une revue des principales techniques d'insertions.

6.1 Choix de l'impédance par logique floue

Comme nous l'avons déjà mentionné, le contrôle d'impédance est utilisé à la fois dans l'espace libre et dans l'espace contraint. Il est évident que les valeurs d'impédance devraient être différentes dans les deux espaces de travail. Dans l'espace libre, les rigidités équivalantes K_p et K_o, des équations 5.15 et 5.16, devaient être assez élevées pour assurer une bonne poursuite de la trajectoire. Par contre ces mêmes rigidités entraîneraient de grands efforts de contacts dans l'espace contraint.

La logique floue a été utilisée pour améliorer la poursuite de trajectoires dans l'espace libre et pour améliorer la compliance résultante de l'effecteur dans l'espace contraint. Nous utilisons la logique floue dans le choix des différents paramètres de l'impédance linéaire. Les paramètres de l'impédance linéaire varient tout au long de la tâche d'insertion selon des règles que nous avons définies, alors que ceux de l'impédance de rotation restent fixes. Bien que nous ne pouvons prétendre avoir choisi des règles optimales, nous observons une amélioration des performances.

À l'annexe III nous présentons des notions de base nécessaires à la compréhension de la logique floue. Nous retrouvons également un exemple du processus de fuzzification jusqu'au processus de défuzzification.

6.1.1 Applications

Un contrôleur utilisant un système de logique floue (*FLS : Fuzzy Logic System*) s'appelle en anglais *Fuzzy Logic Controller (FLC)*. La figure 6.1 illustre les relations entre les diverses composantes de notre FLC. Notons que dans notre application la défuzzification est faite selon la méthode du centre de gravité.

Il est possible d'obtenir les équations de la pulsation naturelle et du facteur d'amortissement pour un système de second ordre (comme c'est le cas pour l'impédance linéaire 5.15). Les équations 6.1 et 6.2 représentent la pulsation naturelle et le facteur

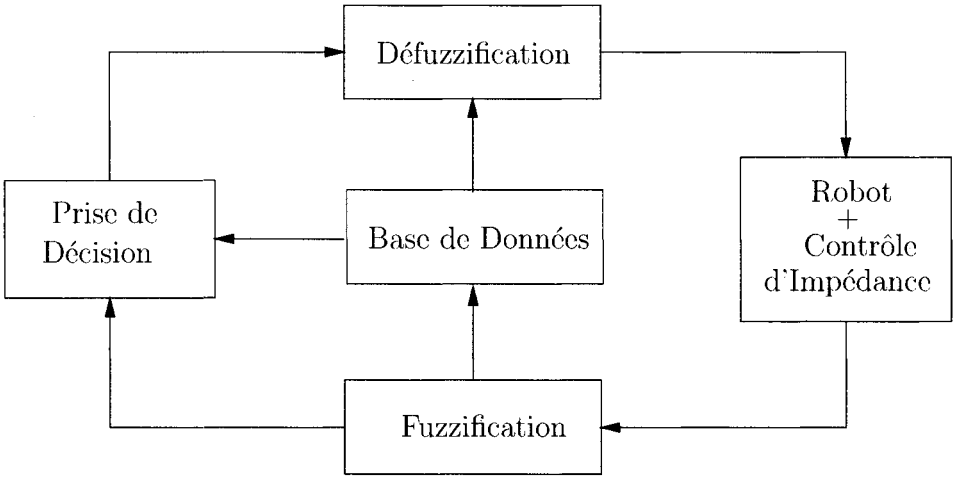


Figure 6.1: Schéma du contrôleur flou

d'amortissement de l'impédance linéaire.

$$\omega_n = \sqrt{\frac{M_{p(i,i)}}{K_{p(i,i)}}} \tag{6.1}$$

$$\zeta = \frac{D_{p(i,i)}}{2\sqrt{M_{p(i,i)}K_{p(i,i)}}} \tag{6.2}$$

Les indices $_{(i,i)}$ dénotent l'élément i,i de la matrice en question. Nous pouvons ainsi ajuster le facteur d'amortissement en variant les éléments de matrices M_p, D_p et K_p de façon appropriée. Notons que ces équations sont valables seulement dans le cas d'impédance ayant une rigidité diagonale et que la réponse du système est libre. Par réponse libre, ou de régulation, nous entendons que

$$\tilde{p} = p_c \tag{6.3}$$

$$\dot{\tilde{p}} = \dot{p}_c \tag{6.4}$$

$$\ddot{\tilde{p}} = \ddot{p}_c \tag{6.5}$$

dans l'équation d'impédance 5.15. Un certain facteur d'amortissement peut donner une réponse acceptable en régime libre. Par contre il se peut fort bien que la réponse en régime exité (lorsque la trajectoire est non nulle) soit moins bonne.

Nous utilisons la logique floue pour faire varier la matrice diagonale de rigidité K_p. Par la suite nous calculons la matrice d'amortissement D_p à partir de M_p et de K_p pour avoir un amortissement donné (d'un système libre). Tout au long de la simulation, la matrice d'inertie M_p est gardée constante.

La variation de la rigidité $k_{p_{3,3}}$ s'effectue selon deux prémices de base. Dans la première, la rigidité varie en fonction de la force exercée par l'effecteur sur l'environnement. Ainsi la rigidité augmente si la force de contact est faible et inversement. Dans la seconde, la rigidité varie selon la différence entre la position en Z actuelle de l'effecteur et la position en Z de la surface à atteindre. Si l'effecteur est loin de l'environnement alors la rigidité sera élevée permettant un meilleur suivi de la trajectoire. Par contre, s'il est proche de l'environnement la rigidité sera faible limitant ainsi la force d'impact.

Les règles floues utilisées sont de la forme suivante :

R_i : si la position en Z est A_1 et la force en Z est B_1 alors $k_{p_{3,3}}$ est C_1

Le tableau 6.1 résume les lois pour la détermination des rigidités $k_{p_{1,1}}$, $k_{p_{2,2}}$ et $k_{p_{3,3}}$. Les abréviations utilisées dans le tableau 6.1 sont définies à la section Listes des Notations et Symboles. Les figures 6.2, 6.3 et 6.4 illustrent les ensembles flous, par rapport au repère de base, reliés à la variation de la rigidité $k_{p_{3,3}}$.

TAB. 6.1 *Règles de logique floue pour* $k_{p_{i,i}}$

Position	Force			
	P	MP	MG	G
TP	TP	TP	TP	TP
P	P	TP	TP	TP
PP	PP	P	TP	TP
MP	MP	PP	TP	TP
MG	MG	MP	TP	TP
PG	PG	MG	P	TP
G	G	PG	PP	P
TG	TG	G	MP	PP

La section 7.2 présente l'effet bénéfique de la variation de l'impédance en z telle que nous l'avons décrit lorsque l'effecteur entre en contact avec l'environnement. On y remarque que la force de contact est plus faible et qu'il y a beaucoup d'oscillations, comme c'est le cas sans logique floue. C'est pour cette raison que Caccavale, Natale, Siciliano et Villani [10] [9] ont choisi les valeurs de D_p fournissant un facteur d'amortissement supérieur à 2. Par contre, un facteur d'amortissement trop élevé détériore la poursuite de trajectoires dans l'espace libre.

Pour réduire ces oscillations, nous multiplions $D_{p3,3}$ par un certain facteur β, qui est obtenu par logique floue. Le but étant de remplacer $D_{p_{3,3}}$ par $\beta D_{p3,3}$ dans l'équation 5.15.

Les rigidités en x ($k_{p_{1,1}}$) et y ($k_{p_{2,2}}$) sont également ajustées par la logique floue, d'un façon similaire à la rigidité en z. Les règles floues ainsi que les ensembles flous sont donnés par le tableau 6.1 et les figures 6.2, 6.7 et 6.8. L'ensemble flou représenté à la figure 6.7 est celui de l'erreur de position par rapport au centre du trou. Plus l'effecteur est loin du trou, du plan parallèlle au plan xy, plus ces rigidités seront élevées. La

section 7.2 présente des résultats lors d'une insertion utilisant cette logique.

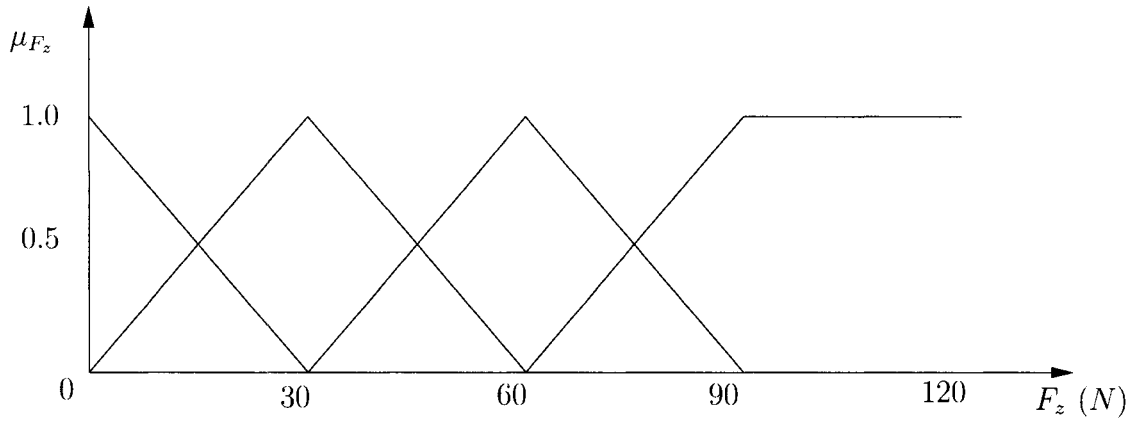


Figure 6.2: Ensemble flou de la force en x, en y et en z

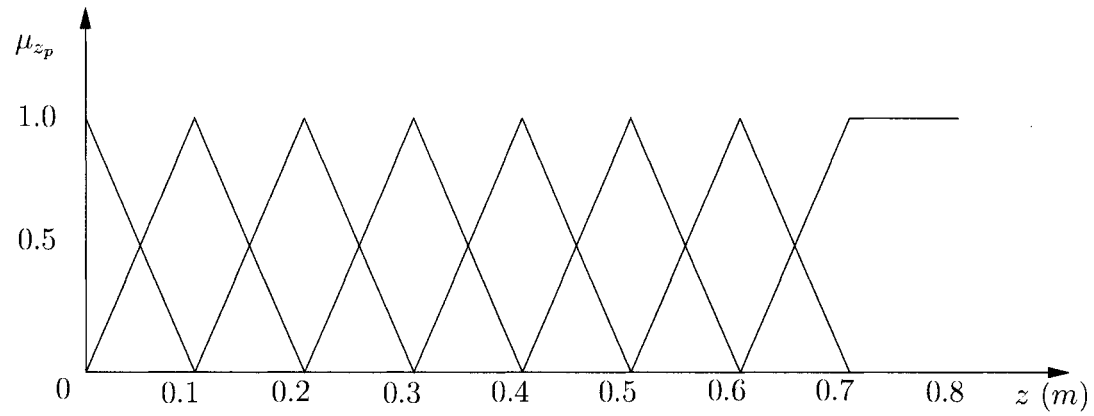


Figure 6.3: Ensemble flou de la position en z utilisé pour la variation de $k_{p_{3,3}}$

6.2 Génération de trajectoires

Dans cette section nous nous intéressons à la génération de trajectoires à l'aide de splines cubiques paramétriques en t pour x, y, z et q, où x, y, z sont des coordonnées linéaires dans le repère de base et q est un quaternion représentant une orientation

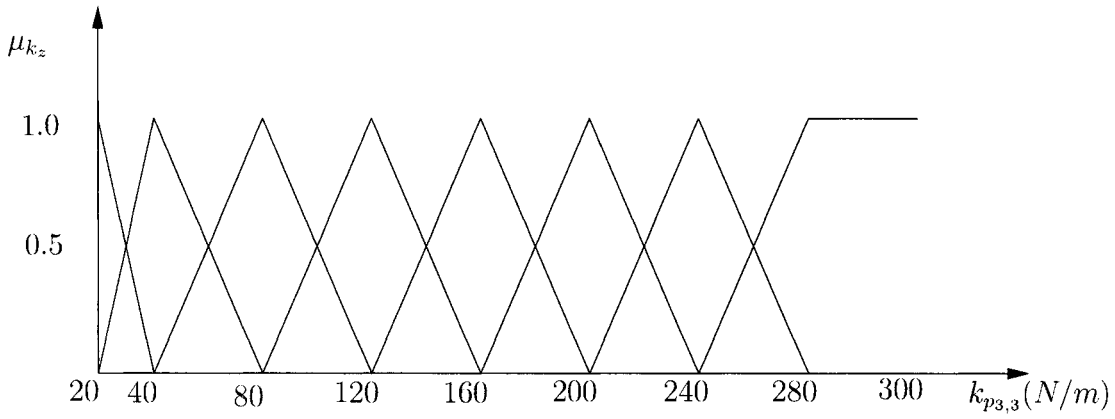


Figure 6.4: Ensemble flou de la rigidité $k_{p_{3,3}}$

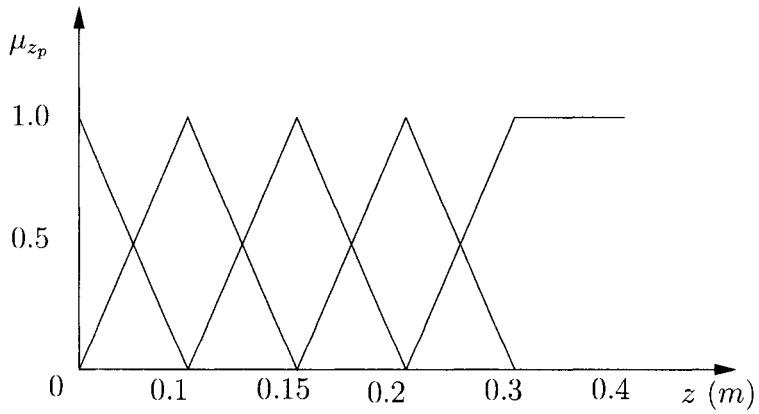


Figure 6.5: Ensemble flou de la position utilisée pour la variation de β

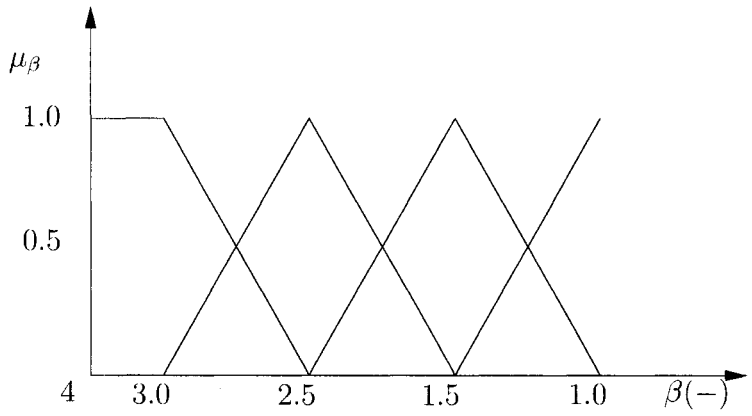


Figure 6.6: Ensemble flou de β

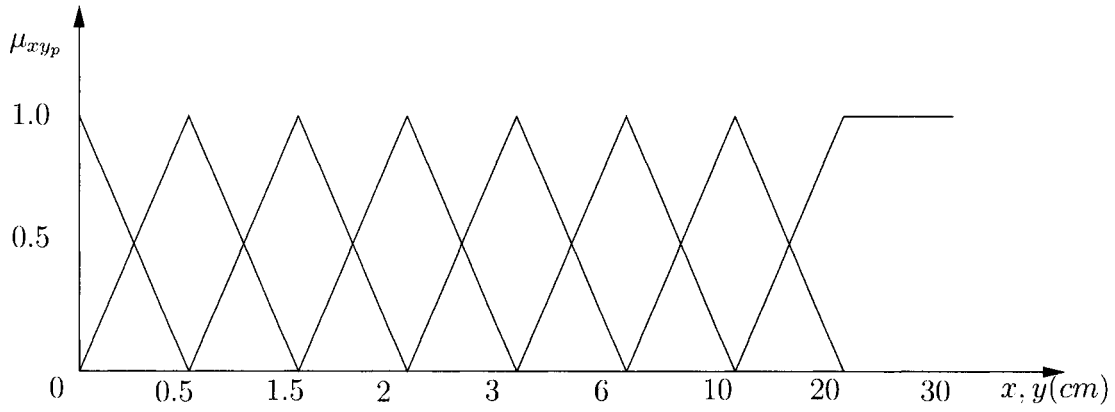


Figure 6.7: Ensemble flou de la position en x et y

dans le repère de base. Les splines cubiques ont l'avantage de fournir des trajectoires (courbes) continues. La position, l'orientation ainsi que leurs dérivées seront utilisées comme entrées dans le contrôleur d'impédance.

Parce que les splines cubiques n'assurent pas la continuité des deuxièmes dérivées aux points de contrôles, l'accélération obtenue par les splines est discontinue. Cet effet indésirable est réduit par l'utilisation d'un plus grand nombre de points de contrôles.

Nous traitons la génération des trajectoires de l'effecteur en deux parties. La

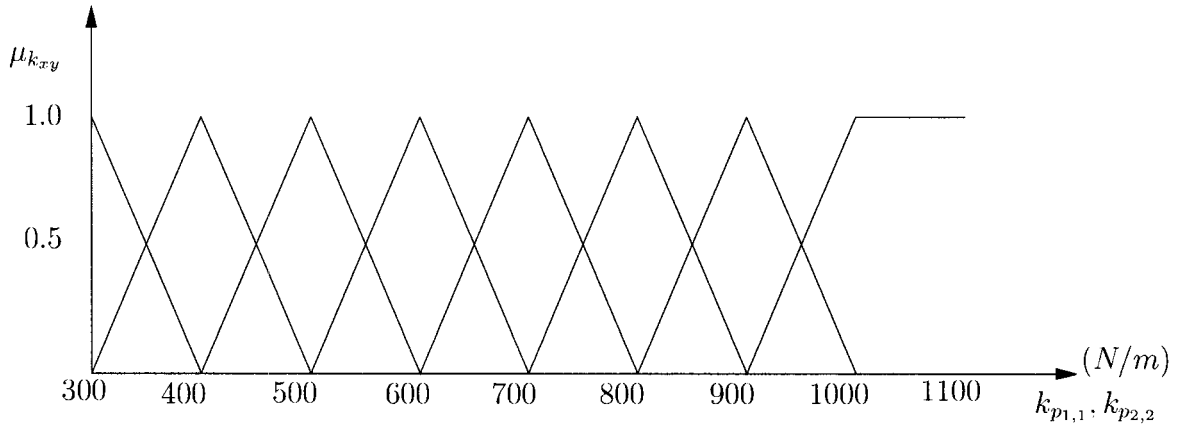


Figure 6.8: Ensemble flou de la rigidité $k_{p_{1,1}}$, $k_{p_{2,2}}$

première est celle des trajectoires de translation dans le repère de base et la seconde est celle des trajectoires de l'orientation autour des axes du repère de base.

6.2.1 Génération des trajectoires de translation par splines cubiques

Nous utilisons des splines cubiques naturelles pour générer des trajectoires dans l'espace cartésien [3]. Considérons deux points d'interpolation consécutifs $P_k(t_k, (x_k, y_k, z_k))$ et $P_{k+1}(t_{k+1}, (x_{k+1}, y_{k+1}, z_{k+1}))$. Le kième polynôme cubique entre ces deux points $(t_k \leq t \leq t_k)$ est donné par

$$s_k(t) = A_k(t - t_k)^3 + B_k(t - t_k)^2 + C_k(t - t_k) + D_k \tag{6.6}$$

et les coefficients sont donnés par

$$A_k = \frac{1}{6\Delta t_k}(s''_{k+1} - s''_k) \tag{6.7}$$

$$B_k = \frac{1}{2}s''_k \tag{6.8}$$

$$C_k = \frac{\Delta s_k}{\Delta t_k} - \frac{1}{6}\Delta t_k(s''_{k+1} + 2s''_k) \tag{6.9}$$

$$D_k = s_k \tag{6.10}$$

où $\Delta s_k \equiv s_{k+1} - s_k$.

La simulation étant faite en trois dimensions, nous utilisons trois splines cubiques pour générer les composantes x, y et z des trajectoires. Les équations 6.6 à 6.10 sont donc utilisées trois fois. L'équation 6.11 nous permet de résoudre les dérivées secondes aux points d'interpolation déterminant ainsi les coefficients A_k à D_k. Notons que la matrice A est toujours inversible car elle est de dimension $N-2 \times N-2$, où N est le nombre de points d'interpolation.

$$As'' = 6Cs \tag{6.11}$$

$$A = \begin{bmatrix} 2\alpha_{1,2} & \alpha_2 & 0 & \cdots & 0 \\ \alpha_2 & 2\alpha_{2,3} & \alpha_3 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \alpha_{N'''} & 2\alpha_{N''',N''} & \alpha_{N''} \\ 0 & 0 & \cdots & \alpha_{N''} & 2\alpha_{N'',N'} \end{bmatrix} \tag{6.12}$$

$$C = \begin{bmatrix} \beta_1 & -\beta_{1,2} & \beta_2 & 0 & \cdots & 0 & 0 \\ 0 & \beta_2 & -\beta_{2,3} & \beta_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \beta_{N'''} & -\beta_{N''',N''} & \beta_{N''} & 0 \\ 0 & 0 & 0 & \cdots & \beta_{N''} & -\beta_{N'',N'} & \beta_{N'} \end{bmatrix} \tag{6.13}$$

Pour $i, j, k = 1, \cdots, N-1$,

$$\alpha_k \equiv \Delta t_k \tag{6.14}$$

$$\alpha_{i,j} \equiv \alpha_i + \alpha_j \tag{6.15}$$

$$\beta_k \equiv 1/\alpha_k \tag{6.16}$$

$$\beta_{i,j} \equiv \beta_i + beta_j \tag{6.17}$$

$$N' \equiv N - 1 \tag{6.18}$$

$$N'' \equiv N - 2 \tag{6.19}$$

$$N''' \equiv N - 3 \tag{6.20}$$

La génération des trajectoires de translation est mise en oeuvre par les classes *Spl_cubic* et *Spl_path* de *ROBOOP*.

6.2.2 Génération de trajectoires de rotation par splines de quaternions

Avant de commencer la lecture de cette section, nous suggérons au lecteur de lire l'annexe I Introduction aux quaternions.

Il est intéressant de développer une technique d'interpolation de quaternions unitaires, pour obtenir une interpolation continue entre des rotations. L'utilisation des angles d'Euler pour faire une telle interpolation est beaucoup plus complexe. Le problème d'interpolation des quaternions n'est pas simple puisque l'ensemble des quaternions constitue un espace non-Euclidien, ce qui exclut par exemple les méthodes d'interpolation comme les splines [21]. Nous employons celles utilisant les fonctions *Slerp* et *Squad*.

Rappelons que le groupe des rotations peut être projeté sur une sphère unitaire, à quatre dimensions, de quaternions unitaires. Une interpolation linéaire entre deux quaternions (*Lerp*) est illustrée à la figure 6.9b). Cette interpolation ne reste pas sur la sphère unitaire des quaternions unitaires, mais la traverse pour se trouver sur une droite. Ce raccourci résulte en une vitesse angulaire non constante le long de l'interpolation entre deux quaternions. La droite est séparée en quatre segments égaux donnant des angles d'interpolation différents. Les quaternions résultants de cet interpolation linéaire ne sont plus unitaires.

L'interpolation linéaire sphérique de quaternions (*Slerp*) interpole les quaternions sur une courbe suivant un arc sur la sphère des quaternions unitaires. Cette méthode est illustrée à la figure 6.9c).

Le quaternion $q(t)$ interpole les quaternions q_0 et q_1 selon le paramètre t sur la

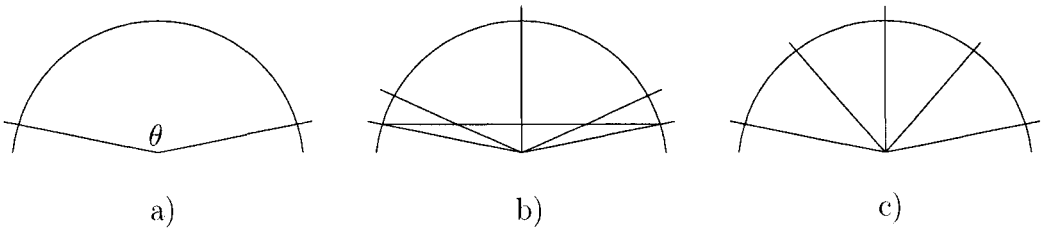


Figure 6.9: Une illustration de la différence entre *Lerp* et *Slerp*. a) L'interpolation couvre l'angle θ. b) Lerp: La droite est séparée en quatre parties égales. c) Slerp: L'angle θ est séparé en quatre parties égales.

sphère des quaternions unitaires. L'idée [45] est donc $q(t) = c_0(t)q_0 + c_1(t)q_1$, où c_0 et c_1 sont des fonctions réelles pour $0 \leq t \leq 1$ ayant comme valeurs limites $c_0(0) = 1$, $c_1(0) = 0$, $c_0(1) = 0$ et $c_1(1) = 1$. Lorsque t varie uniformément entre 0 et 1, $q(t)$ doit varier uniformément sur l'arc de cercle de q_0 vers q_1. Le cosinus de l'angle entre $q(t)$ et q_0 est $\cos(t\theta)$ et celui entre $q(t)$ et q_1 est $\cos((1-t)\theta)$. Les équations pour l'angle entre $q(t)$ et q_0 et celle entre $q(t)$ et q_1 sont respectivement $\cos(t\theta) = c_0(t) + \cos(\theta c_1(t))$ et $\cos((1-t)\theta) = \cos(\theta c_0(t)) + c_1(t)$. On obtient donc les coefficients c_0 et c_1

$$c_0(t) = \frac{\sin((1-t)\theta)}{\sin\theta} \tag{6.21}$$

$$c_1(t) = \frac{\sin(t\theta)}{\sin\theta} \tag{6.22}$$

L'interpolation *Slerp* est définie comme étant

$$Slerp(q_0, q_1, t) = \frac{q_0 \sin((1-t)\theta) + q_1 \sin(t\theta)}{\sin(\theta)} \tag{6.23}$$

pour $0 \leq t \leq 1$. Cette expression peut se réduire dans le cas de quaternions unitaires. L'idée est $q_1 = q_0(q_0^{-1}q_1)$, où le terme $q_0^{-1}q_1 = \cos(\theta) + \hat{u}\sin(\theta)$. L'angle entre les

quaternions q_0 et q_1 est θ. Le paramètre t peut être ajouté à l'angle θ pour faire varier q_0 uniformément sur la sphère entre q_0 et q_1. Donc on obtient

$$\begin{aligned} Slerp(q_0, q_1, t) &= q_0\Big(\cos(t\theta) + \hat{u}\sin(t\theta)\Big) \\ &= q_0\Big(\cos(\theta) + \sin(\theta)\Big)^t \\ &= q_0(q_0^{-1}q_1)^t \end{aligned} \tag{6.24}$$

Les dérivées première et seconde de 6.24 par rapport à t sont une application simple de l'équation I.36 de l'annexe I.

$$\begin{aligned} \frac{d}{dt}Slerp(q_0, q_1, t) &= \frac{d}{dt}q_0(q_0^{-1}q_1)^t \\ &= q_0(q_0^{-1}q_1)^t \; ln(q_0^{-1}q_1) \\ Slerp(q_0, q_1, t)' &= Slerp(q_0, q_1, t)ln(q_0^{-1}q_1) \end{aligned} \tag{6.25}$$

Les quaternions q_1 et $-q_1$ représentent la même rotation. Par contre les valeurs de $Slerp(q_0, q_1, t)$ et $Slerp(q_0, -q_1, t)$ sont différentes. Nous utilisons le plus petit angle entre q_0 et q_1, comme dans le cas de l'erreur d'orientation I.4, pour éviter des rotations supplémentaires.

La courbe d'interpolation générée par *Slerp* est équivalente à une ligne droite. Quelques problèmes se posent en interpolant sur une série de rotations : ni la courbe ni la vitesse angulaires ne sont continues aux points de contrôles. Pour résoudre ces problèmes il faut simplement utiliser une méthode cubique.

Une interpolation cubique peut être obtenue par trois interpolations *Slerp*. Sup-

posons quatre quaternions q_i, q_{i+1}, s_i et s_{i+1} formant les sommets d'un quadrilatère. Interpolons q_i et q_{i+1} pour obtenir a et interpolons s_i et s_{i+1} pour obtenir b. Finalement interpolons a et b pour obtenir c. Cette courbe d'interpolation, appelée *Squad* (Sperical et quadrangle), est définie comme [45] [21]

$$Squad(q_i, s_i, s_{i+1}, q_{i+1}, t) = Slerp(Slerp(q_i, q_{i+1}, t), Slerp(s_i, s_{i+1}, t), 2t(1-t)) \quad (6.26)$$

où s_i et s_{i+1} sont des quaternions intermédiaires donnés par

$$s_i = q_i exp\Big(-\frac{ln(q_i^{-1}q_{i+1}) + ln(q_i^{-1}q_{i-1})}{4}\Big) \quad (6.27)$$

Montrons que cette courbe est continue et différentiable [21]. Il faut d'abord s'assurer que les points de contrôles ont les mêmes valeurs en utilisant deux intervalles consécutifs : $Squad(q_{i-1}, s_{i-1}, s_i, q_i, 1) = Squad(q_i, s_i, s_{i+1}, q_{i+1}, 0)$:

$$\begin{aligned} Squad(q_{i-1}, s_{i-1}, s_i, q_i, 1) &= Slerp(Slerp(q_{i-1}, q_i, 1), Slerp(s_{i-1}, s_i, 0), 0) \\ &= Slerp(q_i, s_i, 0) \\ &= q_i \end{aligned}$$

$$\begin{aligned} Squad(q_i, s_i, s_{i+1}, q_{i+1}, 0) &= Slerp(Slerp(q_i, q_{i+1}, 0), Slerp(s_i, s_{i+1}, 0), 0) \\ &= Slerp(q_i, s_i, 0) \\ &= q_i \end{aligned}$$

Donc une courbe d'interpolation *Squad* est continue et possède la bonne valeur aux points de contrôles. Montrons maintenant, de la même façon, que la dérivée première est également continue :

$$\frac{d}{dt}Squad(q_{i-1}, s_{i-1}, s_i, q_i, 1) = \frac{d}{dt}Squad(q_i, s_i, s_{i+1}q_{i+1}, 0) \tag{6.28}$$

Soit $g_i(t) = Slerp(q_i, q_{i+1}, t)^{-1}Slerp(s_i, s_{i+1}, t)$ un quaternion unitaire,

$$\begin{aligned}
\frac{d}{dt}Squad(q_i, s_i, s_{i+1}, q_{i+1}, t) &= \frac{d}{dt}Slerp(Slerp(q_i, q_{i+1}, t), Slerp(s_i, s_{i+1}, t), 2t(1-t)) \\
Squad(q_i, s_i, s_{i+1}, q_{i+1}, t)' &= \Big(\frac{d}{dt}Slerp(q_i, q_{i+1}, t)\Big)g_i(t)^{2t(1-t)} \\
&= \Big(\frac{d}{dt}Slerp(q_i, q_{i+1}, t)\Big)g_i(t)^{2t(1-t)} + \\
&\quad Slerp(q_i, q_{i+1}, t)\Big(\frac{d}{dt}g_i(t)^{2t(1-t)}\Big) \\
&= Slerp(q_i, q_{i+1}, t)ln(q_i^{-1}, q_{i+1})g_i(t)^{2t(1-t)} + \\
&\quad Slerp(q_i, q_{i+1}, t)\Big(\frac{d}{dt}g_i(t)^{2t(1-t)}\Big)
\end{aligned} \tag{6.29}$$

Évaluons maintenant la dérivée de *Squad* aux points de contrôles afin de déterminer une expression pour les quaternions intermédiaires, s_i et s_{i+1}, et ce en utilisant les

équations 6.29 et I.38

$$\begin{aligned} Squad(q_{i-1}, s_{i-1}, s_i, q_i, 1)' &= Slerp(q_{i-1}, q_i, 1) ln(q_{i-1}^{-1} q_i) + \\ & \quad Slerp(q_{i-1}, q_i, 1) \frac{d}{dt}\Big(g_{i-1}(t)^{2t(1-t)}\Big)(1) \\ &= q_i\Big(ln(q_{i-1}^{-1} q_i) - 2ln(g_{i-1}(1))\Big) \\ &= q_i\Big(ln(q_{i-1}^{-1} q_i) - 2ln(q_i^{-1} s_i)\Big) \end{aligned} \tag{6.30}$$

$$\begin{aligned} Squad(q_i, s_i, s_{i+1}, q_{i+1}, 0)' &= Slerp(q_i, q_{i+1}, 0) ln(q_i^{-1} q_{i+1}) + \\ & \quad Slerp(q_i, q_{i+1}, 0) \frac{d}{dt}\Big(g_i(t)^{2t(1-t)}\Big)(0) \\ &= q_i\Big(ln(q_i^{-1} q_{i+1}) - 2ln(g_i(0))\Big) \\ &= q_i\Big(ln(q_i^{-1} q_{i+1}) + 2ln(q_i^{-1} s_i)\Big) \end{aligned} \tag{6.31}$$

Les équations 6.31 et 6.32 étant égales, on obtient après simplification l'équation générale 6.27 pour les points de contrôles intermédiaires s_i. Ainsi nous avons montré que *Squad* est continue et différentiable aux points de contrôles intermédiaires s_i sur tous les segments.

L'algorithme de l'expression *Squad* permet d'obtenir une courbe d'interpolation pour une série de quaternions $q_0, q_1, \ldots, q_n$. L'expression *Squad* n'est pas définie sur le premier et sur le dernier intervalle car q_{-1} apparaît dans l'expression de s_0, et q_{n+1} apparaît dans l'expression de s_n. Sur les premier et dernier segments on utilise une interpolation *Slerp*. En choisissant q_0, q_1 et q_{n-1}, q_n proche les uns des autres on

minimise l'effet de l'utilisation d'une interpolation *Slerp* au lieu d'une interpolation *Squad.*

Sur tous les intervalles nous calculons également la dérivée de l'interpolation, soit celle de *Slerp* pour les premier et dernier intervalles, soit celle de *Squad* pour les autres. De cette dérivée et de la loi de propagation des quaternions I.24, nous obtenons la vitesse angulaire dans le repère de base. Ainsi nous obtenons des trajectoires de position angulaire et de vitesse angulaire, qui sont fournies au contrôleur. La figure 6.10 illustre la vitesse angulaire, obtenue à l'aide de la loi de propagation des quaternions, pour la spline de quaternions ayant les points de contrôles du tableau 6.2.

La dérivée de *Slerp* est constante puisqu'il s'agit d'une méthode de premier ordre. C'est pourquoi les composantes de la vitesse angulaire sont constantes sur le premier intervalle, 0 jusqu'à 0.5 seconde. À 0.5 seconde il y a une discontinuité due au fait que l'interpolation est maintenant réalisée avec la fonction *Squad.* Plus les temps supérieurs à 0.5 seconde les composantes sont continues. Étant donné que l'interpolation *Squad* est de troisième ordre, on remarque une discontinuité dans les dérivées à chaque transition d'intervalle.

temps (sec)	quaternions
0.0	{0, [-1,0,0]}
0.5	{0.019, [0.976,0.194,0.102]}
2.0	{0.335, [-0.928,-0.078,-0.140]}
2.5	{0.327, [-0.886,-0.241,-0.222]}
16.5	{0.461, [-0.601,-0.643,0.112]}

TAB. 6.2 *Points de contrôles d'une spline de quaternions*

La génération de trajectoire d'orientation est implantée dans la classe *Spl_Quaternion*

sous *ROBOOP*.

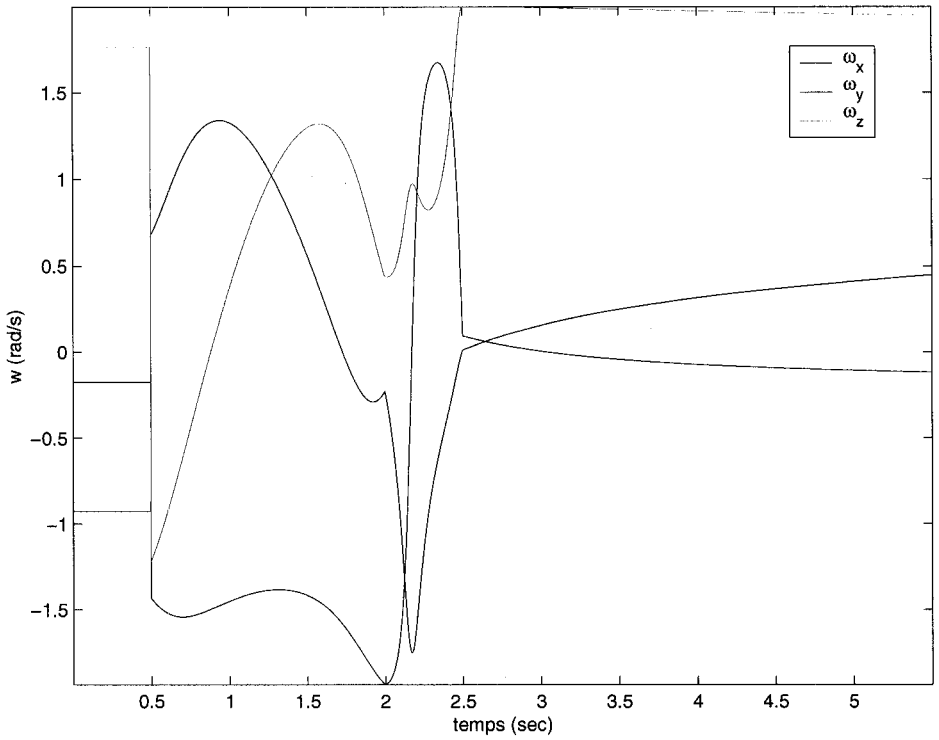


FIG. 6.10 – Vitesse angulaire, ω, obtenue par spline de quaternions.

6.3 Recherche du trou

Rappelons que notre but est d'effectuer l'insertion automatique d'une tige cylindrique dans un trou cylindrique par un robot manipulateur. La tâche à réaliser est simple pour un humain mais plus complexe pour un robot. Nous utilisons une technique de recherche pour compenser les imprécisions dans notre connaissance de l'environnement. Chhatpar et Branicky [12] ont montré que dans l'assemblage de deux pièces cylindriques de transmission automobile, que l'insertitude radiale de positionnement des pièces est de plus de six fois le jeu d'assemblage (clearance) entre les deux

pièces.

Il y a principalement deux types de recherche possible, soit celle utilisant la vision artificielle et celle aveugle. La vision artificielle est un outil puissant mais complexe et coûteux. Dans beaucoup d'assemblages, le jeu entre les pièces est moins de $0.1mm$, alors que la résolution d'une caméra typique est d'environ $1mm$ [43]. De plus, dans certains assemblages, comme celui des pièces de transmission mentionné ci-haut, la technique de vision artificielle ne peut pas être utilisée puisqu'une partie des pièces est cachée durant l'assemblage. Comme il a été mentionné au chapitre 1, nous n'utilisons pas de technique de vision artificielle dans cette étude.

Un humain aveugle représente un bon modèle à suivre pour la modélisation de la recherche. Il ne connait pas la position du trou avec certitude. Il tente un premier essai pour insérer la tige dans le trou. S'il échoue, il recherche le trou et une fois trouvé, il procède à l'insertion. La recherche et l'insertion sont traitées aux sections 6.3.1 et 6.4 respectivement.

Quelle genre de recherche doit-on faire ? Deux techniques ont été étudiées jusqu'à présent, soit la recherche en zig zag et la recherche en spirale. La recherche en zig zag n'optimise pas l'aire de recherche [43]. Lors de la recherche en spirale, seulement une faible partie de la spirale sera parcourue si le départ est près du trou. Par contre lorsque le départ de la recherche est éloigné du trou, la technique en zig zag pourrait s'avérer plus efficace. Chhatpar et Branicky [12] ont discrétisé l'aire de recherche pour déterminer les paramètres de la spirale à utiliser.

6.3.1 Discrétisation de l'aire de recherche

Nous utiliserons une technique similaire à celle employée dans le chapitre 4 pour discrétiser l'espace de recherche.

Au lieu de vérifier si la tige est contenue dans le trou, vérifions que le centre de la tige est contenu dans un cercle de rayon β, où β est le jeu fonctionnel entre la tige et le trou tel qu'illustré à la figure 6.11 [12].

La centre de la tige n'a pas besoin de balayer tout l'espace de recherche H. Il suffit de vérifier que le centre de la tige est à une distance plus petite ou égale à β du centre du trou. L'espace de recherche est donc discrétisé par le jeu fonctionnel, β.

Si nous divisons l'espace de recherche en cercles concentriques espacés de 2β radialement, alors nous sommes assurés que n'importe quel points de l'espace de recherche est à une distance maximale de β d'un de ces cercles. La figure 6.12 illustre la trajectoire en cercles concentriques. Cette trajectoire est uniquement utilisée pour définir la trajectoire en spirale.

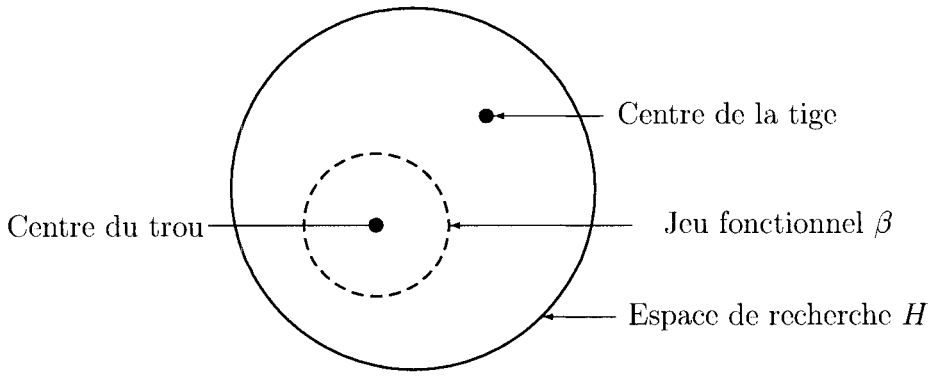


FIG. 6.11 Description de l'espace de recherche

Les dimensions de la spirale sont déterminées par l'espace de recherche H et le jeu

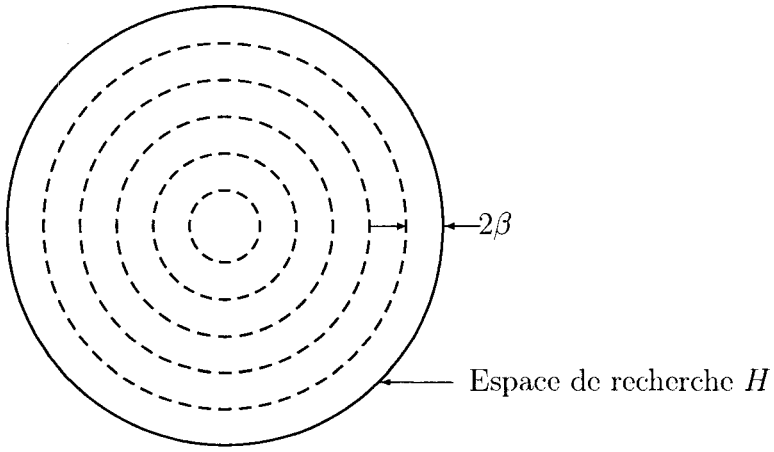


FIG. 6.12 Discrétisation de l'espace de recherche en cercles concentriques.

fonctionnel β.

$$\frac{\Delta r}{\Delta \theta} = \frac{\beta}{2\pi} \tag{6.32}$$

$$\dot{r} = \frac{\beta}{2\pi}\dot{\theta} \tag{6.33}$$

En utilisant les équations 6.32, 6.33 ainsi que les équations 6.34 à 6.38, on peut obtenir la vitesse angulaire $\dot{\theta}$ définissant une vitesse tangentielle constante tout au long

de la trajectoire 6.39.

$$x = r\cos(\theta) \tag{6.34}$$

$$y = r\sin(\theta) \tag{6.35}$$

$$\dot{x} = -r\sin(\theta)\dot{\theta} + \cos(\theta)\dot{r} \tag{6.36}$$

$$\dot{y} = r\cos(\theta)\dot{\theta} + \sin(\theta)\dot{r} \tag{6.37}$$

$$\dot{s} = v = \sqrt{\dot{x}^2 + \dot{y}^2} = \sqrt{r^2\dot{\theta}^2 + \dot{r}^2} \tag{6.38}$$

$$\dot{\theta} = \frac{v}{\sqrt{r^2 + \frac{\beta^2}{4\pi^2}}} \tag{6.39}$$

La figure 6.13 illustre un exemple de trajectoire de recherche en spirale. Le départ de la spirale (le centre) commence lorsque la tige entre en contact avec l'environnement et se termine par l'insertion de la tige dans le trou.

La recherche en spirale est activée lorsque l'effecteur semble près de la cible et qu'une force verticale (dans l'espace cartésien) est détectée par le capteur de force situé à l'effecteur. La génération de la trajectoire en spirale utilise les splines cubiques paramétriques de la section 6.2.1 ainsi que les splines cubiques de quaternions de la 6.2.2.

Les deux principales limitations que nous avons remarquées dans l'utilisation de cette technique sont la vitesse de déplacement dans la spirale et la possibilité que le robot atteigne une position de singularité (le trou est hors de portée du robot). Le suivi de la trajectoire en spirale sera moins précis avec une vitesse élevée de l'effecteur.

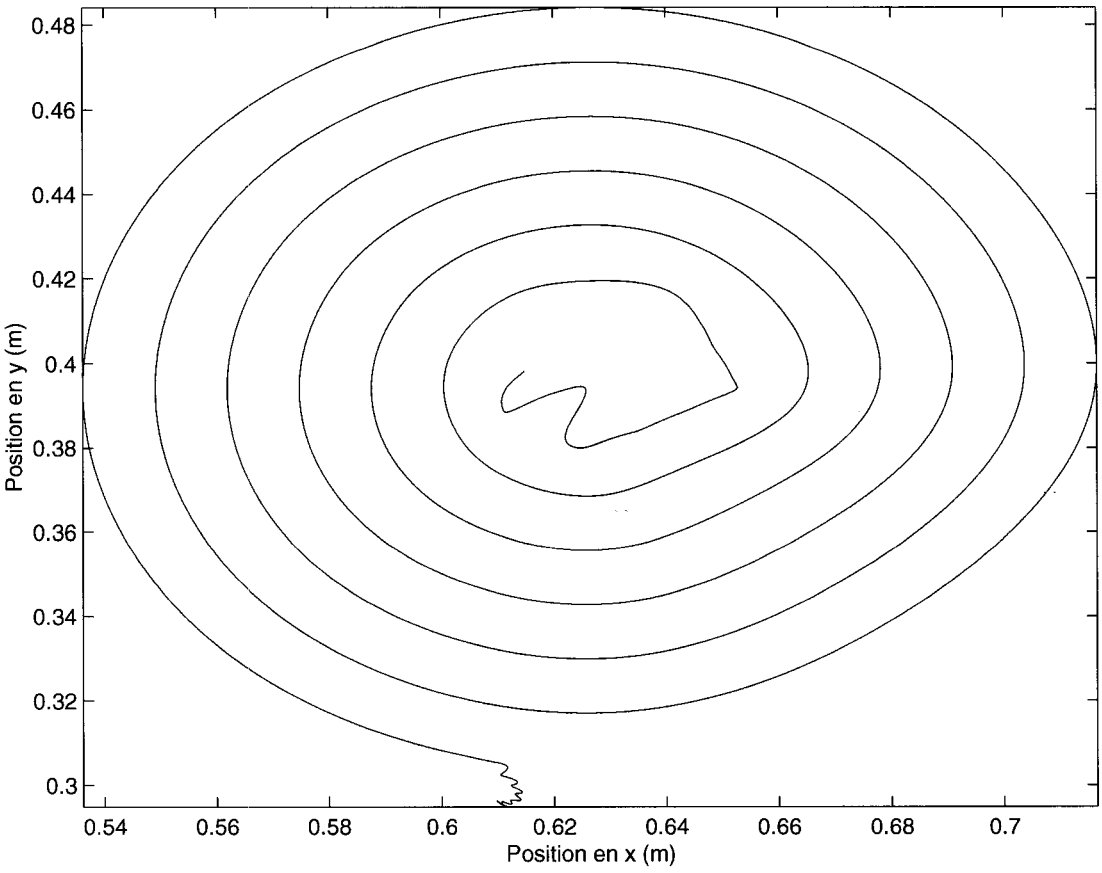


FIG. 6.13 – Exemple de recherche en spirale, $\beta = 1.5mm$ et $H = 0.5m$.

De plus, il risque d'y avoir des dommages structuraux (robot, tige et trou) lors de l'insertion de la tige dans le trou. L'évitement d'une région singulière peut se faire en utilisant la technique de l'inverse de la Jacobienne selon les moindres carrés amortis ou par la technique alternative présentée à la section 5.4.

6.3.2 Trou incliné

Au chapitre 4, nous avons fait l'hypothèse que la surface du trou se trouvait dans un plan parallèle au plan X_bY_b du repère de base. Que faire si la surface du trou est maintenant contenue dans le $X'Y'$ (plan ayant subi des rotations de ϕ autour de X_b et ψ autour de Y_b) ? Cette sous-section propose une solution au problème sans toutefois faire de simulation.

La solution envisagée est de trouver l'équation du plan, contenant la surface du trou, par la méthode des moindres carrés. Une fois que le nouveau plan est trouvé, il suffit de positionner l'axe axial de l'effecteur parallèle à la normale du plan $X'Y'$.

Voici les équations permettant d'identifier le nouveau plan [27]. Généralement l'équation d'un plan s'écrit sous la forme $Ax + By + Cz + D = 0$. Réécrivons cette équation sous la forme d'un modèle de régression linéaire

$$z = \beta_0 + \beta_1 x + \beta_2 y + \epsilon \tag{6.40}$$

où β_0 est l'intersection avec l'axe de z, β_1 et β_2 représente les coefficients partiels de régression. En effet, β_1 mesure le changement en z par unité de changement de x en gardant y constant et β_2 mesure le changement en z par unité de changement de y en

gardant x constant.

Les moindres carrés peuvent être utilisés pour estimer les coefficients de régression. Pour ce faire on doit avoir plus de trois triplets de points (x, y, z). Notons par (x_i, y_i, z_i) le ième triplet de points, et assumons que l'erreur ϵ dans le modèle a une espérance nulle, $E(\epsilon) = 0$, et une variance $V(\epsilon) = \sigma^2$ et que les ϵ_i ne sont pas corrélées.

La somme des moindres carrés est

$$\begin{aligned} L &= \sum_{i=i}^{n} \epsilon_i^2 \\ &= \sum_{i=1}^{n} \Big(z_i - \beta_0 - \beta_1 x_i - \beta_2 y_i \Big)^2 \end{aligned} \tag{6.41}$$

La somme L doit être un minimum par rapport à β_0, β_1 et β_2. L'estimateur des moindres carrés doit satisfaire

$$\frac{\partial L}{\partial \beta_0} = -2 \sum_{i=1}^{n} \Big(z_i - \hat{\beta}_0 - \hat{\beta}_1 x_i - \hat{\beta}_2 y_i \Big) = 0 \tag{6.42}$$

$$\frac{\partial L}{\partial \beta_1} = -2 \sum_{i=1}^{n} \Big(z_i - \hat{\beta}_0 - \hat{\beta}_1 x_i - \hat{\beta}_2 y_i \Big) x_i = 0 \tag{6.43}$$

$$\frac{\partial L}{\partial \beta_2} = -2 \sum_{i=1}^{n} \Big(z_i - \hat{\beta}_0 - \hat{\beta}_1 x_i - \hat{\beta}_2 y_i \Big) y_i = 0 \tag{6.44}$$

En simplifiant ces équations on obtient

$$n\hat{\beta}_0 + \hat{\beta}_1 \sum_{i=1}^{n} x_i + \hat{\beta}_2 \sum_{i=1}^{n} y_i = \sum_{i=1}^{n} z_i \tag{6.45}$$

$$\hat{\beta}_0 \sum_{i=1}^{n} x_i + \hat{\beta}_1 \sum_{i=1}^{n} x_i^2 + \hat{\beta}_2 \sum_{i=1}^{n} x_i y_i = \sum_{i=1}^{n} x_i z_i \tag{6.46}$$

$$\hat{\beta}_0 \sum_{i=1}^{n} y_i + \hat{\beta}_1 \sum_{i=1}^{n} x_i y_i + \hat{\beta}_2 \sum_{i=1}^{n} y_i^2 = \sum_{i=1}^{n} y_i z_i \tag{6.47}$$

$$\tag{6.48}$$

La solution des équations précédentes est l'estimation des moindres carrés des coefficients de régression $\hat{\beta}_0$, $\hat{\beta}_1$ et $\hat{\beta}_2$. Il est plus facile de résoudre ces équations si elles sont exprimées sous forme matricielle $z = X\beta + \epsilon$

$$L = \sum_{i=1}^{n} \epsilon_i^2 = \epsilon'\epsilon = \big(z - X\beta\big)'\big(z - X\beta\big) \tag{6.49}$$

L'estimateur des moindres carrés doit satisfaire

$$\frac{\partial L}{\partial \beta} = -2X^T z + 2X^T X\hat{\beta} = 0 \tag{6.50}$$

L'estimateur $\hat{\beta}$ est donc

$$\hat{\beta} = X^+ z \tag{6.51}$$

où $X^+ = (X^T X)^{-1} X^T$ est la pseudo inverse de X.

Il est possible de déterminer les angles d'inclinaison du plan de travail par rapport

au repère de base. Ainsi en utilisant la normale au plan, $N_p = \beta_1\vec{i} + \beta\vec{j} - \vec{k}$, on obtient

$$\phi = \cos^{-1}\left(\frac{N_p \cdot \vec{i}}{|N_p||\vec{i}|}\right) \tag{6.52}$$

$$\theta = \cos^{-1}\left(\frac{N_p \cdot \vec{j}}{|N_p||\vec{j}|}\right) \tag{6.53}$$

$$\psi = \cos^{-1}\left(\frac{N_p \cdot \vec{k}}{|N_p||\vec{k}|}\right) \tag{6.54}$$

6.4 Insertion de la tige

Pour des raisons de simplicité, nous avons considéré que l'axe longitudinal de la tige est parrallèle à l'axe du trou lors de l'approche finale. C'est ce que Pettinaro appelle la technique *straight thrusting* [43].

Une brève revue de certaines techniques explorées est présentée. Nous pouvons classer ces technique en deux catégories, soit celle de recherche du trou et celle aidant à l'insertion une fois que le trou est trouvé.

Lors d'une insertion, un humain incline la pièce à insérer pour pallier au mauvais alignement de la pièce et du trou [6]. De plus, cette technique peut être facilement combinée à la recherche en spirale pour améliorer les chances de trouver le trou [12].

En pratique, l'insertion directe, *straight thrusting*, doit être utilisée avec précaution, à cause du risque de coincement. Pettinaro [43] a mis en oeuvre deux autres techniques pour faciliter l'insertion d'une tige dans deux trous en tandem. La première appelée *wobbling* consiste à faire tourner la tige autour de l'axe du trou lors de l'insertion.

Cette technique est bien entendu valide seulement pour des tiges et des trous de formes cylindriques. La seconde approche, qui semble plus prometteuse que la première, dans les cas d'insertions avec trous en tandem, consiste à déplacer la tige légèrement à l'intérieur du premier trou pour aligner correctement les deux trous.

Une technique intéressante, développée par Naghdy et Nguyen, est d'utiliser la logique floue pour redresser la tige, de façon compliante, lors du début de l'insertion [37].

Pour essayer de valider certaines de ces techniques en simulation, nous aurions dû améliorer le modèle dynamique entre la tige et le trou. Notre modèle semble correct lorsque la surface de la tige est complètement contenue à l'intérieur du trou. Nous n'avons pas considéré les insertions partielles, c'est-à-dire une partie de la surface de la tige est dans le trou alors qu'une autre est à l'extérieur. De plus il faudrait simuler les effets de coincement pour pouvoir comparer ces techniques entre elles. Nous aimerions essayer la technique de Naghdy et Nguyen lors d'une prochaine recherche.

CHAPITRE 7

ANALYSE PAR SIMULATION

Ce chapitre présente différents résultats de simulation. La section 7.1 traite de la performance de l'observateur, la section 7.2 illustre l'effet bénifique de l'ajout de l'ajustement de certains paramètres de l'impédance de translation par logique floue et finalement, on retrouve à la section 7.3 une discution portant sur l'ajout d'un capteur de forces et moments à la base, du robot, pour estimer les forces et les moments à l'effecteur.

Notons que l'on fait référence aux tableaux 3.2 et 3.4 lorsque nous mentionnons que l'observateur connaît parfaitement le modèle du robot, alors que l'on fait référence aux tableaux 3.3 et 3.5 lorsque nous mentionnons que l'observateur ne connaît pas parfaitement le modèle.

7.1 Analyse de Observateur

Cette section porte sur l'analyse de l'observateur décrit à la section 5.5. Nous comparons ses performances d'une part lorsqu'il connait parfaitement la dynamique du robot et d'autre part lorsqu'il la connait avec une certaine erreur. Pour ce faire nous avons utilisé deux instances des classes de robot durant nos simulations. La première *mRobotgl* simule le *Puma 560*. C'est également cette classe qui fait l'animation du manipulateur en *OpenGL*. La deuxième instance, qui est de la classe *mRobot*, a été utilisée pour simuler ce que l'observateur connait du robot. En connaissance parfaite

du modèle, les paramètres du manipulateur seront les mêmes dans les deux classes. De la même façon, les paramètres des deux classes seront différents pour simuler une connaissance imparfaite du robot. Rappelons que les paramètres, exactes et ceux entachés d'erreurs, du robot se trouvent à la section 3.3.

Nous avons fait suivre au robot une trajectoire quelconque, illustrée à la figure 7.1, pour vérifier que l'observateur estime de façon approprié les vitesses angulaires des joints. Il est noté que le contrôleur n'utilisait pas les données fournies pas l'observateur.

Comme on devait s'y attendre, l'observateur estime parfaitement les vitesses angulaires lorsqu'il connait parfaitement le modèle dynamique du robot. L'erreurs d'estimation des vitesses angulaire est nulle, tel qu'illustré à la figure 7.2. Par contre, il en est autrement lorsque l'observateur ne possède pas le modèle exact du manipulateur. Les figures 7.3 et 7.4 montrent l'erreur sur l'estimation des vitesses angulaires de l'observateur. La première figure est générée pour les gains $k_d = 0$ et $k_{p_{ii}} = 0$ alors que la seconde l'est pour les gains $k_d = 20$, $k_{p_{ii}} = 400$. Les gains font référence aux équations 5.57 à 5.59.

Les figures 7.5 a) et b) illustrent les erreurs de suivi des trajectoires de translation et de rotation pour un contrôleur d'accélération résolue pour un observateur connaissant parfaitement le modèle du robot, alors que la figure 7.6 a) et b) illustrent ces erreurs pour un observateur ne connaissant pas parfaitement le modèle du robot. Les figures 7.7 a) et b) et 7.8 a) et b) montrent les erreurs pour un contrôleur de couples précalculés.

L'erreur d'orientation est représentée de deux façons, soit par l'angle extrait du quaternion d'erreur d'orientation et par la norme de la partie vectorielle de ce même

quaternion. L'erreur d'orientation est exprimée par la partie vectorielle du quaternion d'erreur. Les erreurs de positions et d'orientations sont un peu plus petites pour le contrôleur d'accélération résolue. Probablement que l'ajout de l'algorithme *CLIK* au contrôleur de couples précalculés contribue à augmenter les erreurs de positions et d'orientations. De plus, nous avons remarqué qu'il est possible d'utiliser un pas de calcul plus grand en utilisant l'approche par couples précalculés. Il serait intéressant dans une autre étude de vérifier ce point.

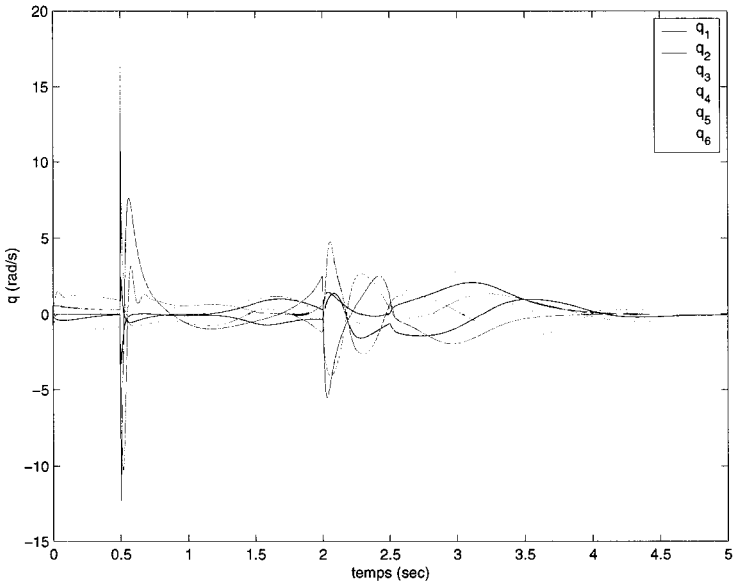


FIG. 7.1 – Vitesses angulaires réelles du robot.

7.2 Analyse de la logique floue

Cette section présente quelques résultats pour montrer l'effet bénéfique de l'utilisation de la logique dans l'ajustement des paramètres de l'impédance de translation. Nous utilisons les valeurs par défaut, définies à la section 5.4, lorsque la logique floue

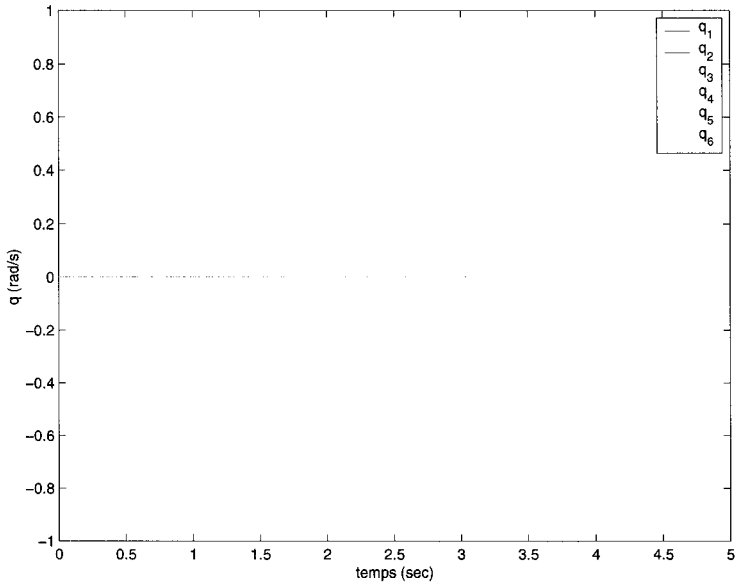


FIG. 7.2 – Erreur d'estimation des vitesses angulaires de l'observateur connaissant parfaitement le modèle du robot.

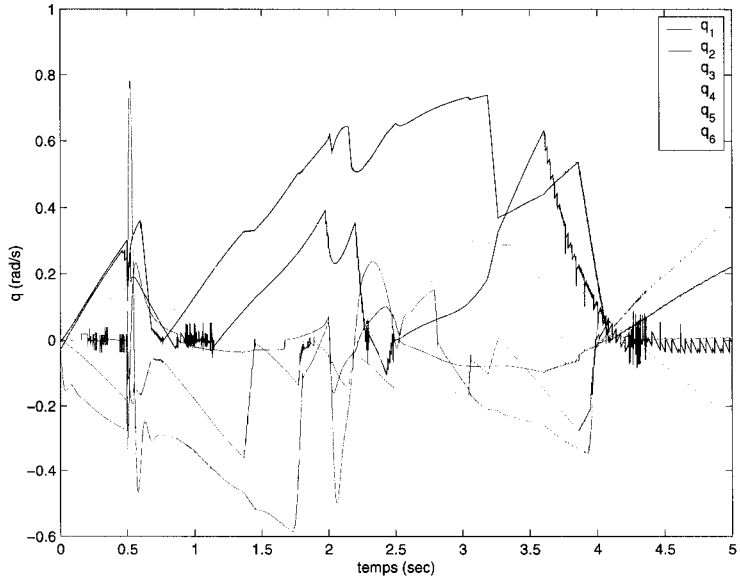


FIG. 7.3 – Erreur d'estimation des vitesses angulaires de l'observateur ne connaissant pas parfaitement le modèle du robot, $K_d = 0$, $k_{p_{ii}} = 0$.

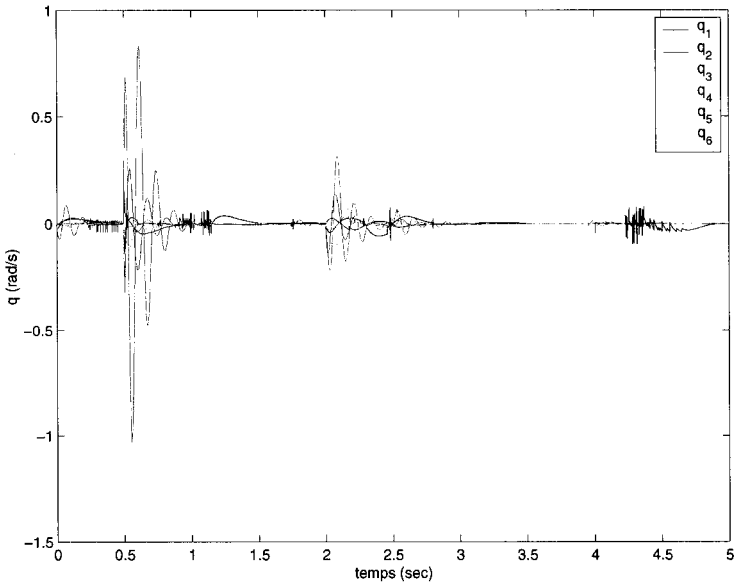
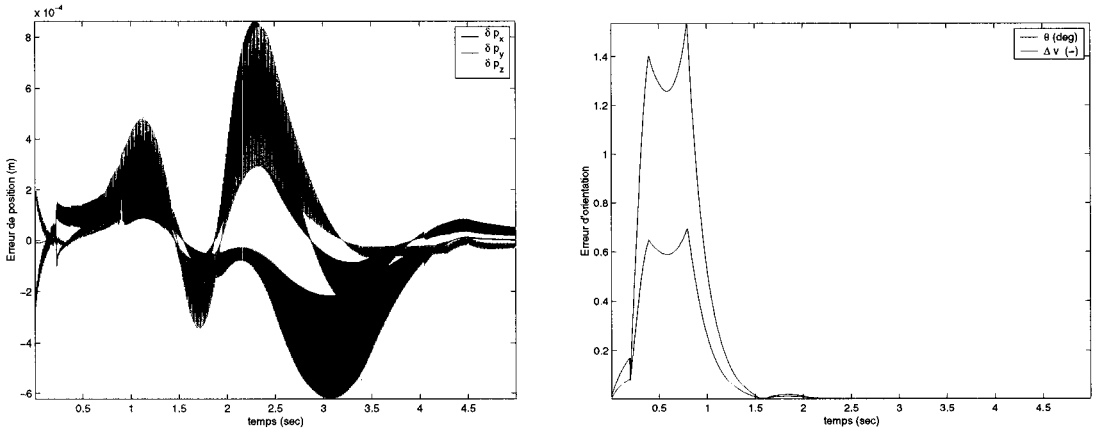
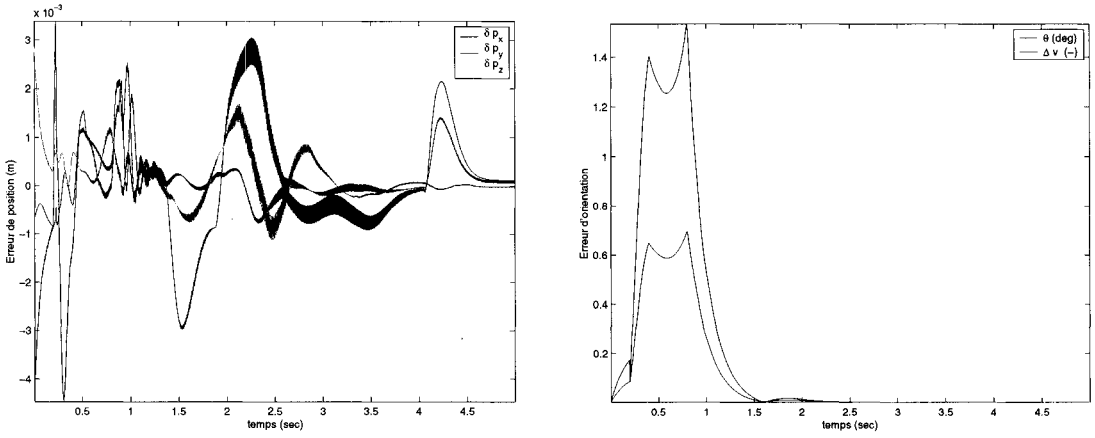


FIG. 7.4 – Erreur d'estimation des vitesses angulaires de l'observateur ne connaissant pas parfaitement le modèle du robot, $K_d = 20$, $k_{p_{ii}} = 400$.



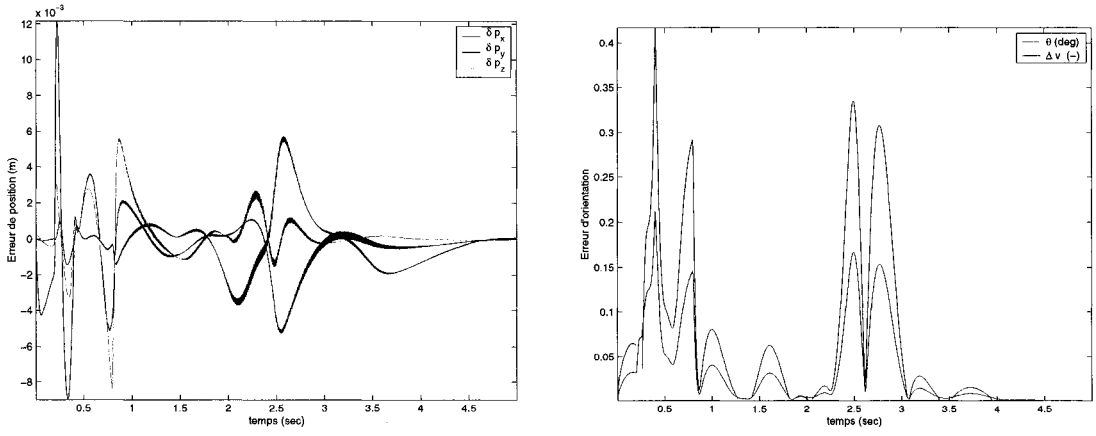
(a) (b)

FIG. 7.5 – Erreur de position et d'orientation obtenues par un contrôleur d'accélération résolue pour un observateur connaissant parfaitement le modèle du robot.



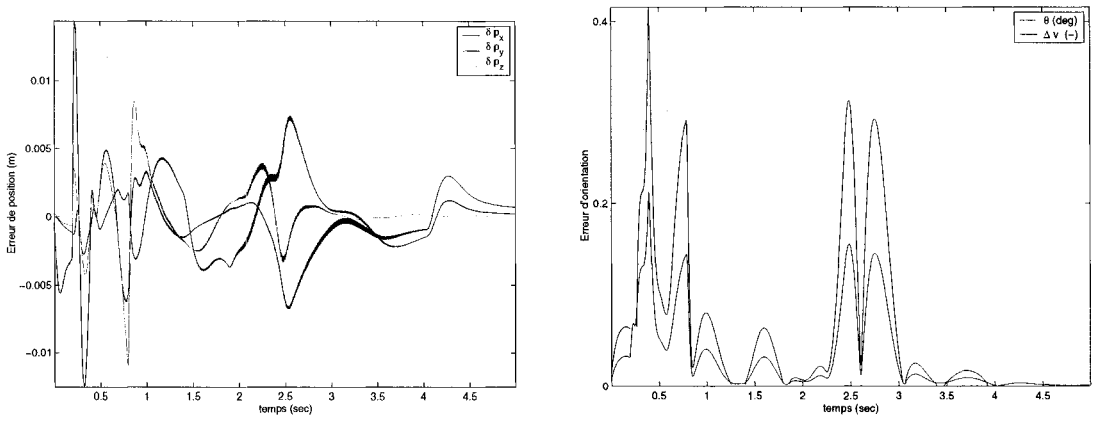
(a) (b)

FIG. 7.6 – Erreur de position et d'orientation obtenues par un contrôleur d'accélération résolue pour un observateur ne connaissant pas parfaitement le modèle du robot.



(a) (b)

FIG. 7.7 – Erreur de position et d'orientation obtenues par un contrôleur de couples précalculés pour un observateur connaissant parfaitement le modèle du robot.



(a) (b)

FIG. 7.8 – Erreur de position et d'orientation obtenues par un contrôleur de couples précalulés pour un observateur ne connaissant pas parfaitement le modèle du robot.

est inactive. Dans nos simulations nous avons utilisé le contrôleur d'impédance avec un contrôle de position basé sur l'accélération résolue accompagné de l'observateur.

Nous avons réalisé deux séries de trois essais dans lesquels l'effecteur évoluait dans l'espace libre pour ensuite entrer en contact avec l'environnement (espace contraint). Dans la première série, l'observateur connaissait parfaitement le modèle du robot, alors que ce n'était pas le cas pour la deuxième série. Les essais devaient suivre la même trajectoire qui était donnée par des splines, tel que décrit à la section 6.2. L'environnement était parallèle au plan X_bY_b du repère de base. La friction n'étant pas simulée, seule une force en z (selon le repère de base) est générée. Le premier essai a été réalisé sans logique floue. Le second utilisait la logique floue pour faire varier la rigidité $K_{p_{i,i}}$ de l'impédance de translation alors que le dernier utilisait la logique floue pour faire varier la rigidité $K_{p_{i,i}}$ et l'amortissement $D_{p_{3,3}}$ (en z_b).

La figure 7.9 montre l'évolution de la position cartésienne de l'effecteur (p_e) et

la position désirée (p_d) pour le troisième essai de la première série. La position du premier et deuxième essai est très similaire à celle du troisième essai. De la même façon, la figure 7.10 illustre la position cartésienne du troisième essai de la deuxième série. La figure 7.11 illustre la force de contact selon l'axe z_b (dans le repère de base) entre l'effecteur et l'environnement pour la première série. On remarque dans les deux derniers essais que la variation de la rigidité permet de diminuer les efforts de contact. On constate également qu'il y a beaucoup d'oscillations dans les deux premiers essais. L'ajout de la logique floue pour augmenter l'amortissement lorsque l'effecteur est près de l'environnement permet de réduire de beaucoup les oscillations. De façon similaire, la figure 7.12 illustre les résultats du troisième essai de la deuxième série. Les figures 7.13, et 7.14 illustrent, pour le troisième essai de la deuxième série, respectivement la variation de la rigidité et la variation de l'amortissement.

Nous avons réalisé deux essais pour examiner l'effet bénifique de la variation des rigidités $K_{p_{1,1}}$ et $K_{p_{2,2}}$ (en x_b et y_b). La figure 7.15 illustre les forces en x et en y, sans et avec logique floue, lorsque la tige est insérée dans le trou. De son côté la figure 7.16 illustre les forces en x et en y, sans et avec logique floue, lorsqu'un un obstacle se trouve sur la trajectoire (dans l'espace libre) de l'effecteur.

Les figures 7.17 a) à d) illustre une tâche d'insertion complète, c'est-à-dire de l'approche jusqu'à l'insertion.

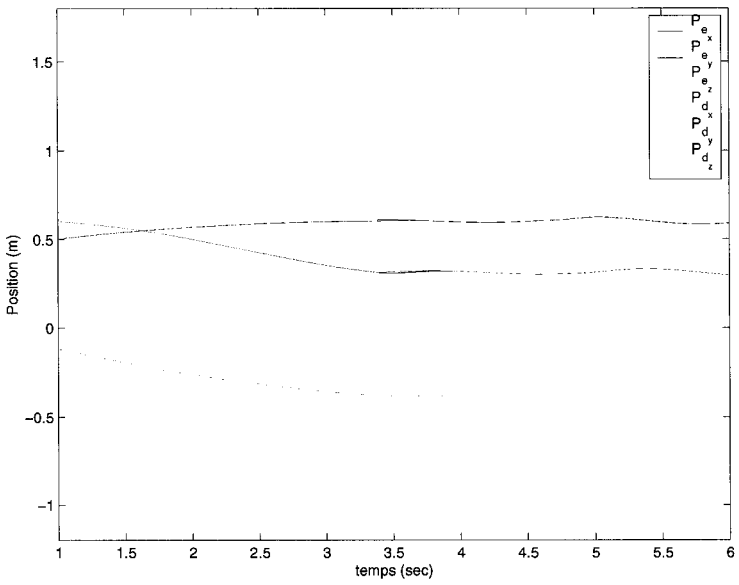


FIG. 7.9 – Position cartésienne de l'effecteur (p_e) et position désirée (p_d) lors d'un contact avec l'environnement en utilisant la logique floue pour un contrôleur connaissant parfaitement le modèle du robot.

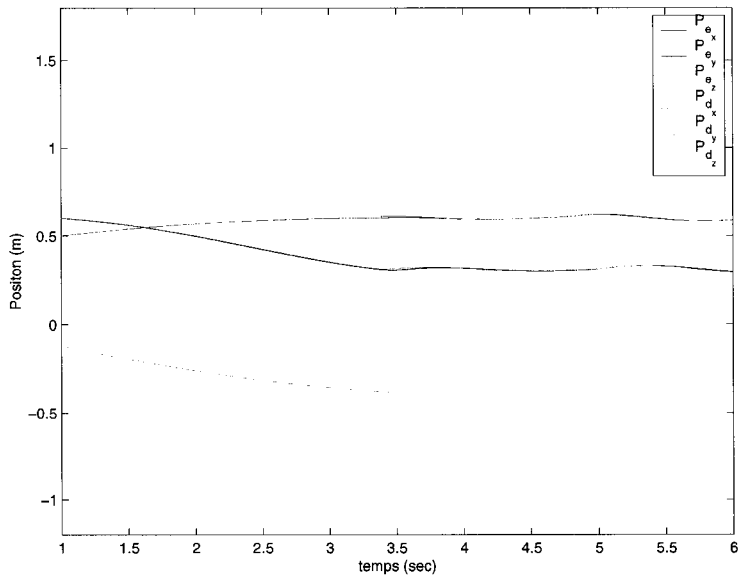


FIG. 7.10 – Position cartésienne de l'effecteur (p_e) et position désirée (p_d) lors d'un contact avec l'environnement en utilisant la logique floue pour un contrôleur ne connaissant pas parfaitement le modèle du robot.

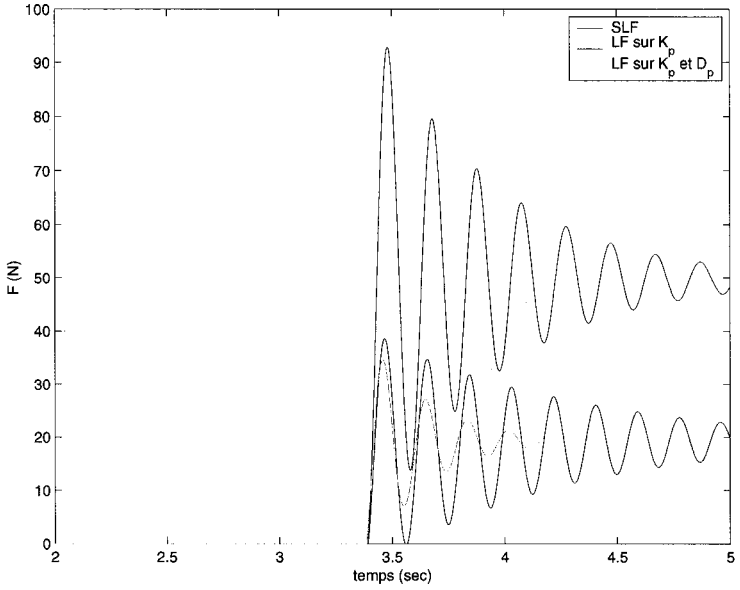


FIG. 7.11 – Comparaison de la force en z lors d'un contact avec l'environnement, sans et avec logique floue pour un contrôleur connaissant parfaitement le modèle du robot.

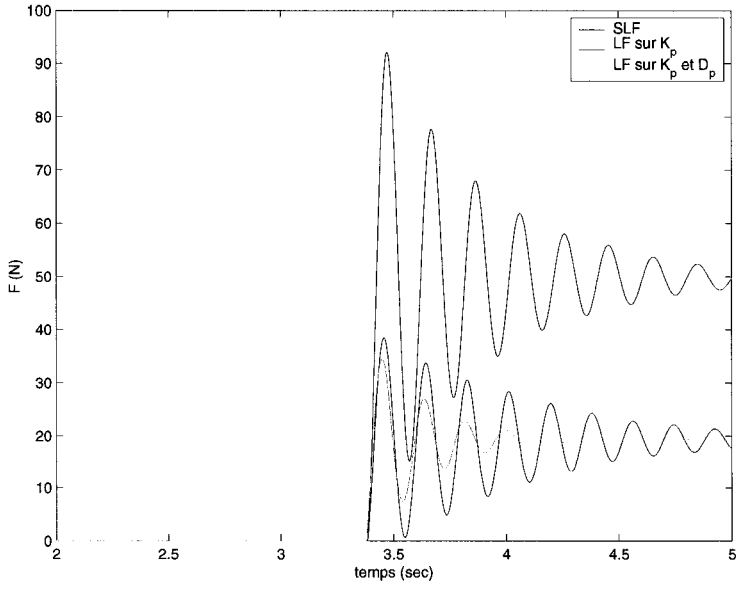


FIG. 7.12 – Comparaison de la force en z lors d'un contact avec l'environnement, sans et avec logique floue pour un contrôleur ne connaissant pas parfaitement le modèle du robot.

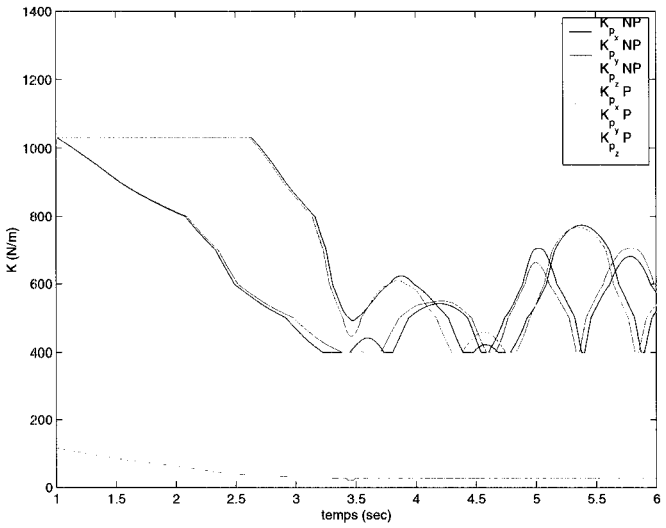


FIG. 7.13 – Variation de la rigidité, K_p, de l'impédance de translation par logique floue lors d'un contact avec l'environnement, pour un contrôleur ne connaissant pas parfaitement (NP) le modèle du robot et un contrôleur connaissant parfaitement (P) le modèle.

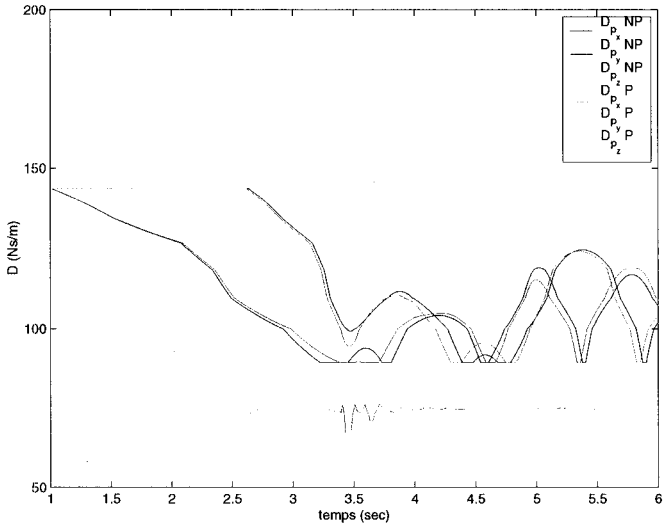


FIG. 7.14 – Variation de la l'amortissement, D_p, de l'impédance de translation par logique floue lors d'un contact en l'environnement, pour un contrôleur ne connaissant parfaitement (NP) le modèle du robot et un contrôleur connaissant parfaitement (P) le modèle.

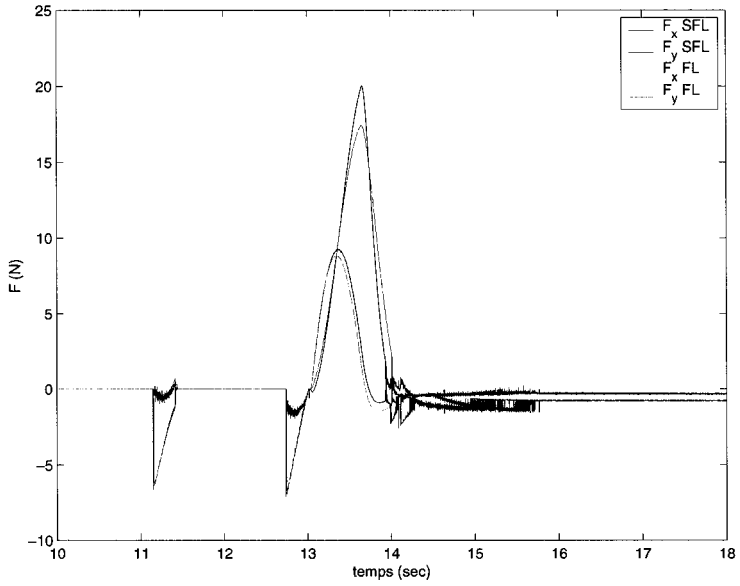


FIG. 7.15 – Comparaison des forces F_x et F_y lors d'une insertion, avec et sans logique floue.

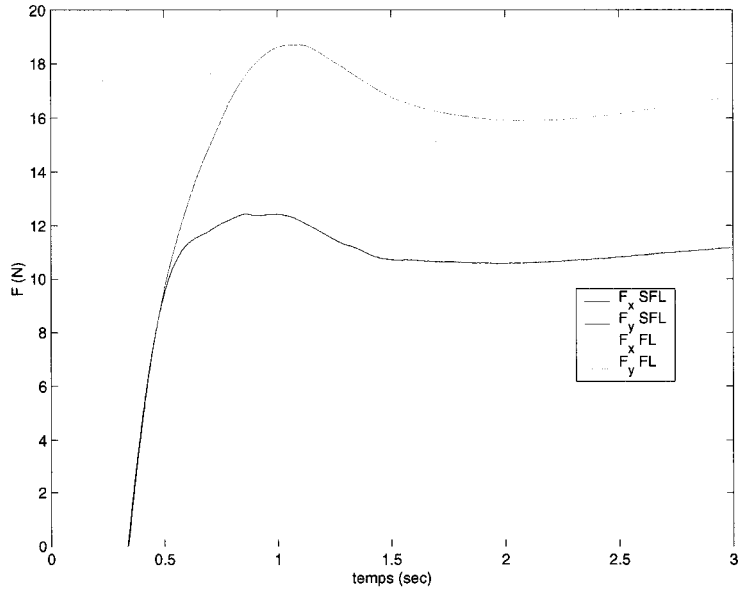
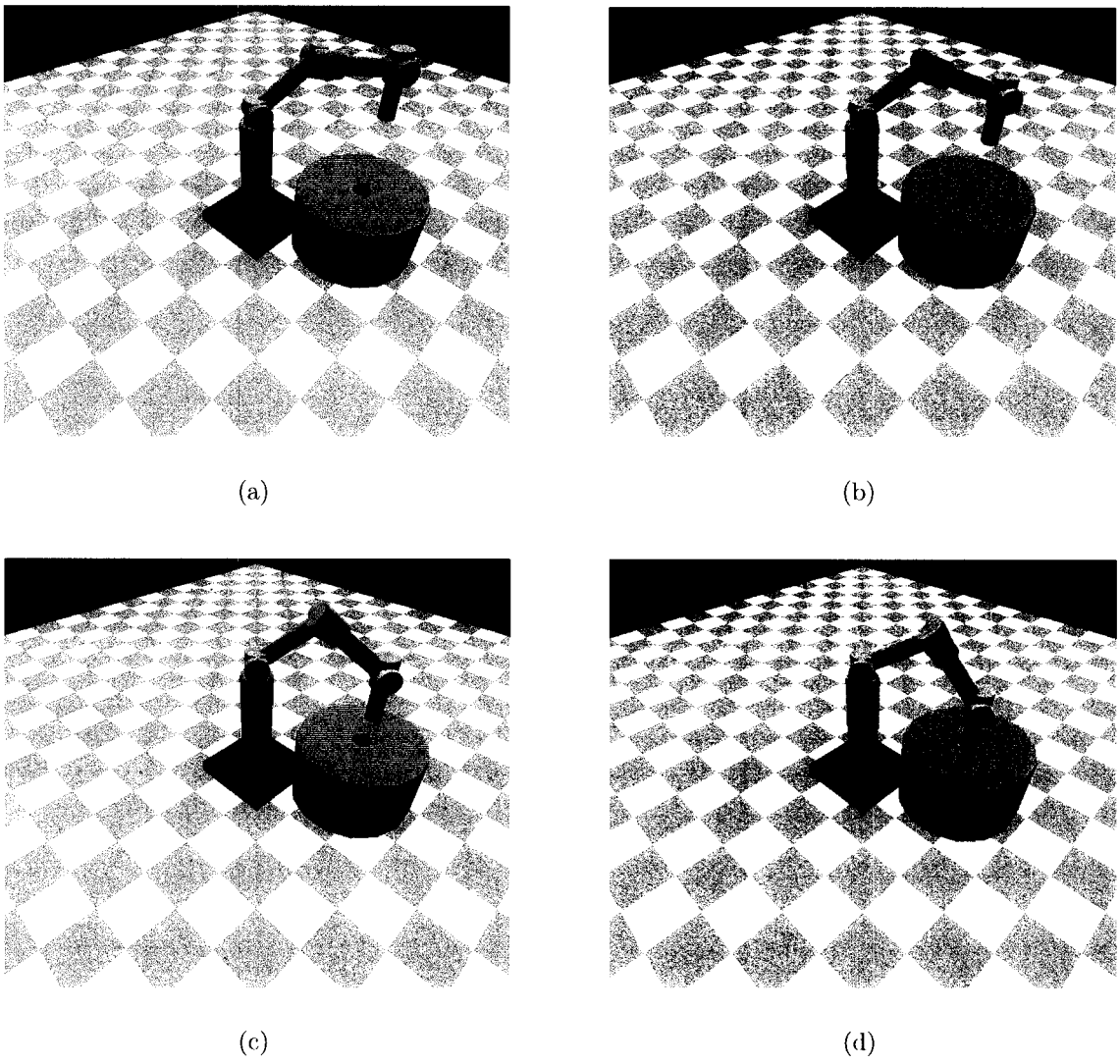


FIG. 7.16 – Comparaison des forces F_x et F_y lors d'un contact avec l'environnement, avec et sans logique floue.



(a) (b)

(c) (d)

FIG. 7.17 – Animation complète de l'insertion.

7.3 Capteur de forces et moments à la base

Jusqu'à présent, dans cette étude, les mesures de forces et de moments sont obtenues par un capteur placé à l'effecteur. Qu'en serait-il si elles étaient obtenues par un capteur situé à la base du robot ?

L'avantage principal d'utiliser un capteur de force à la base serait l'élimination du poids d'un capteur à l'effecteur. Cet avantage est très intéressant pour les applications spatiales. Un autre avantage, montré par Dubowski et Morel [36], est de rendre un contrôle de position assez précis même en présence de friction aux joints.

Serait-il possible d'utiliser un contrôleur d'impédance pour des tâches avec contact, mais en utilisant un capteur à la base plutôt qu'à l'effecteur dans des d'assemblage ou d'insertion ? Connaissant les mesures de forces et de moments, obtenues par le capteur à la base, et la dynamique du robot, on peut estimer les forces et moments à l'effecteur par l'équation suivante

$$F_{eff} = F_{base} - F_{dyn} \tag{7.1}$$

$$N_{eff} = N_{base} - N_{dyn} \tag{7.2}$$

où F et M sont des mesures de forces et moments respectivement, les indices $_{eff}$, $_{base}$ et $_{dyn}$ représente l'effecteur, le capteur à la base et la dynamique du robot respectivement. Les forces et moments sont exprimés dans un même repère, nous avons choisi le repère de base.

Malheureusement cette avenue ne semble pas très fructueuse tant dans un environ-

nement terrestre que dans un environnement spatial. En effet, si le modèle du robot utilisé par le contrôleur est grossièrement estimé, les mesures à l'effecteur seront faussées, et ce même si le manipulateur se déplace lentement. Le contrôleur d'impédance changera alors la position et l'orientation pour diminuer les forces de contact, donc le robot ne suivra pas la trajectoire désirée dans l'espace libre. On serait porté à croire qu'un robot se déplaçant lentement poserait moins de problème qu'un autre se déplaçant à vitesse plus élevée car l'effet de sa dynamique serait plus faible. Nous avons également remarqué de l'instabilité lorsque le manipulateur se déplace à de faibles vitesses. Il semble que l'utilisation d'un capteur de forces et moments à la base d'un robot manipulateur n'est pas une très bonne avenue pour l'estimation des forces et moments à l'effecteur car l'estimation des forces et moments dépendent de l'état du système.

L'agorithme utilisé pour simuler un capteur à la base est le suivant :

1. Simuler le capteur à la base en utilisant les mesures de forces et de moments du premier joint du robot.
2. Calculer les forces et moments à l'effecteur du robot en soustrayant la dynamique du robot estimé.
3. Obtenir les couples au joints déterminés par le contrôleur d'impédance. Le contrôleur utilise le modèle du robot estimé et les forces et moments estimés à l'effecteur.
4. Calculer la dynamique du robot réel et du robot estimé. La dynamique du robot réel utilise les valeurs réelles des forces et des moments de contact, alors que le robot estimé utilise les valeurs estimées. Lorsqu'il n'y a pas d'observateur les positions et vitesses angulaires sont identiques pour les deux robots.

Nous avons fait trois essais, avec gravité, avec gravité sans observateur et sans gravité, pour démontrer notre hypothèse lorque l'effecteur évolue dans l'espace libre. Nous avons utilisé le contrôleur d'impédance accompagné du contrôleur de position d'accélération résolue.

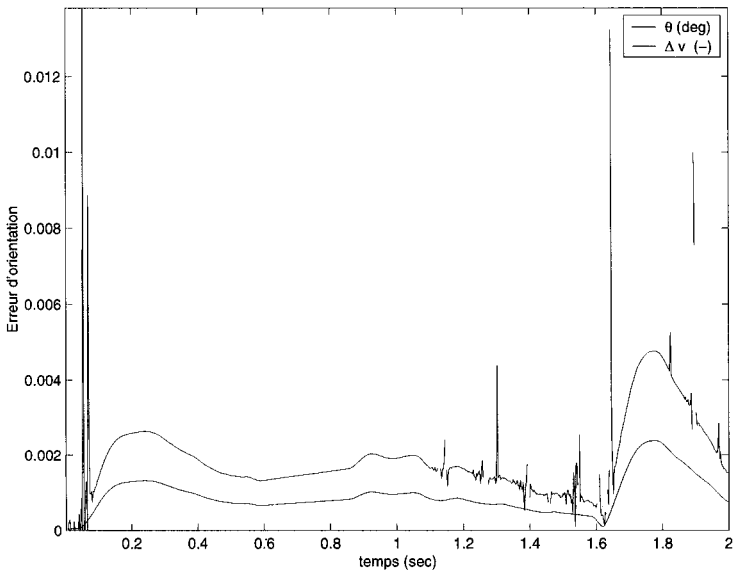


FIG. 7.18 – Erreur d'orientation dans un environnement terrestre sans capteur à la base.

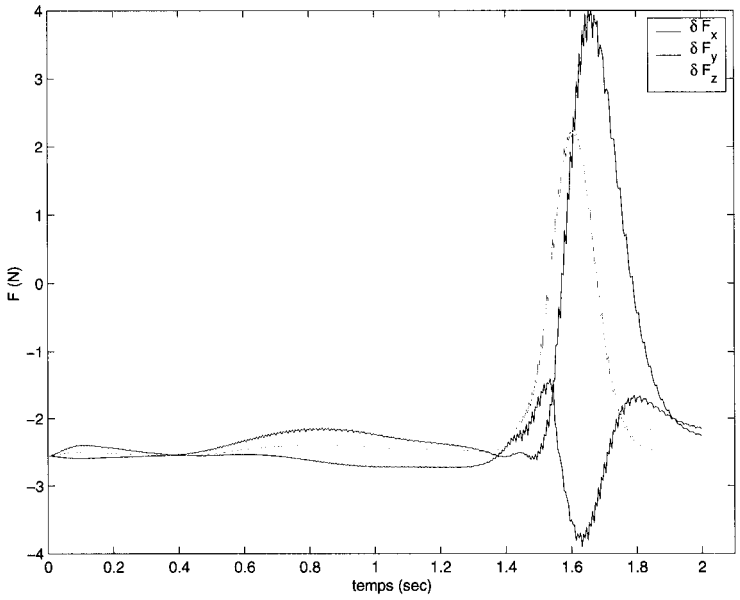


FIG. 7.19 – Erreur de la force estimée dans un environnement terrestre avec un capteur à la base.

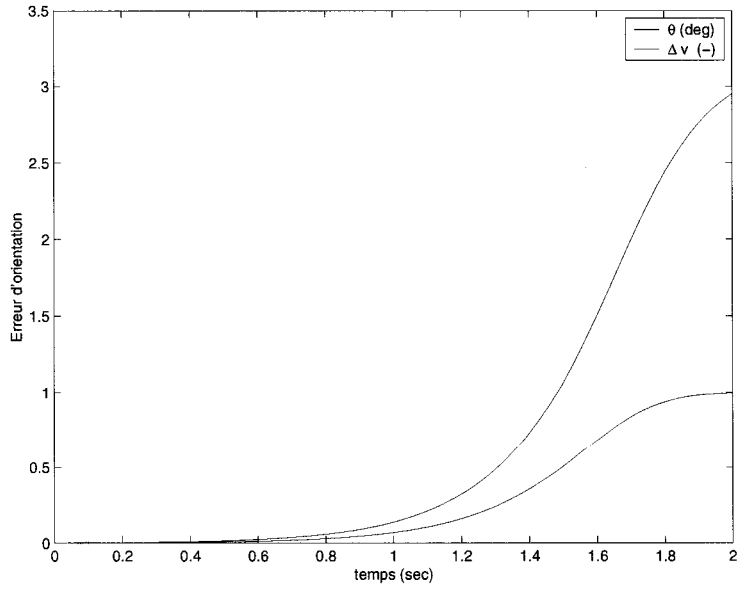


FIG. 7.20 – Erreur d'orientation dans un environnement terrestre avec un capteur à la base.

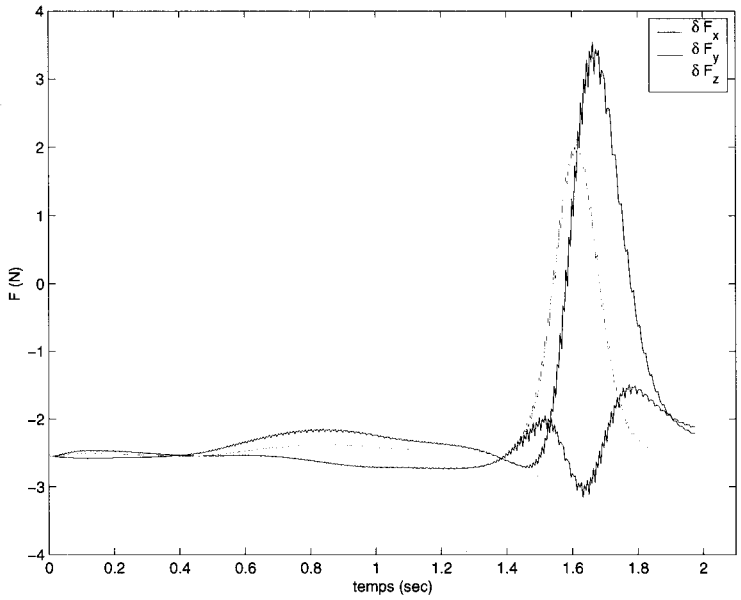


FIG. 7.21 – Erreur de la force estimée dans un environnement terrestre avec un capteur à la base sans observateur.

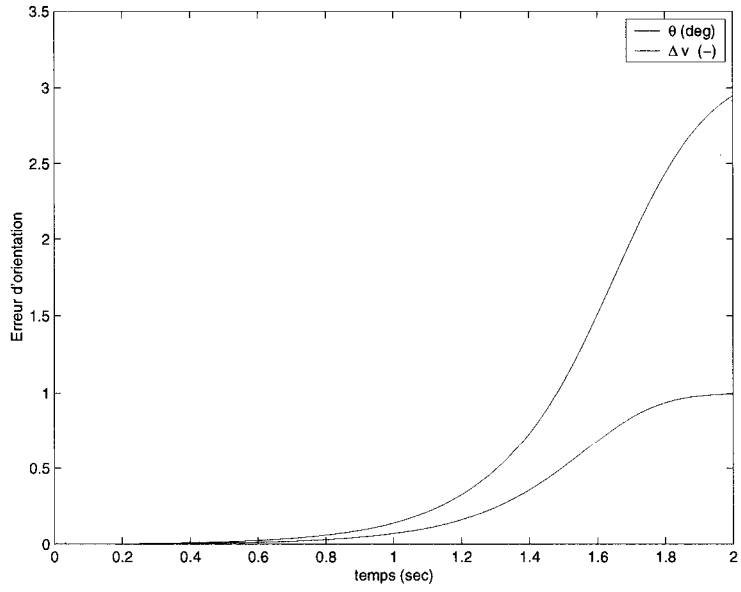


FIG. 7.22 – Erreur d'orientation dans un environnement terrestre avec un capteur à la base sans observateur.

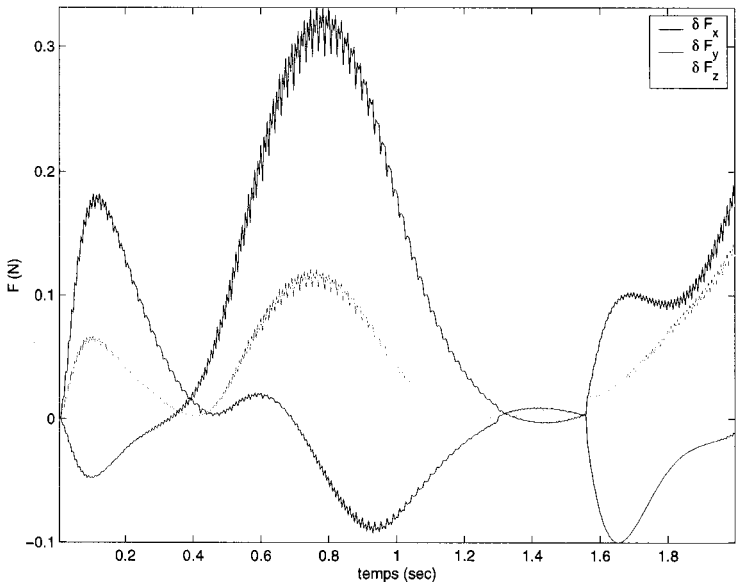


FIG. 7.23 – Erreur de la force estimée dans un environnement spatial avec un capteur à la base.

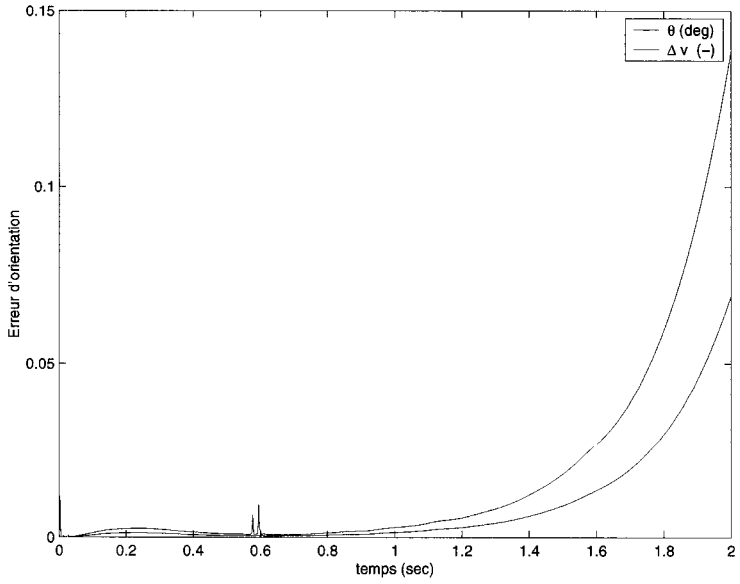


FIG. 7.24 – Erreur d'orientation dans un environnement spatial avec un capteur à la base.

CONCLUSION

Dans cette étude nous avons décris les composantes nécessaires pour la conception d'un simulateur d'une tâche d'insertion par un robot manipulateur. Un des buts que nous nous étions fixés était d'utiliser un robot le plus économique possible. Ainsi le manipulateur ne possède pas de système de vision artificielle ni de tachymètre (capteur de vitesse des joints).

Nous avons choisi un contrôleur d'impédance, ayant une matrice de rigidité diagonale, pour guider le robot tant dans l'espace libre que dans l'espace contraint. Rappelons que ce contrôleur ne devient pas instable durant la transition de l'espace libre à l'espace contraint. Le contrôleur se divise en deux parties, soit la génération de trajectoires compliantes par les impédances de translation et de rotation et le suivi de ces trajectoires par un contrôleur de position. Nous avons fait l'analyse en utilisant les contrôleurs de position selon l'accélération résolue et selon la méthode du couple précalculé. Le premier est plus simple et donne de meilleurs résultats.

Un observateur non linéaire des vitesses angulaires a été utilisé pour pallier au fait que notre robot ne possede pas de tachymètres. Nous avons démontré sous quelles conditions les agencements de l'observateur et les contrôleurs d'accélération résolue et de couples précalculés sont stables.

L'utilisation de la logique floue, pour la modification des paramètres de l'impédance de translation, a permis d'améliorer les performances du contrôleur surtout lorsque l'effecteur entre en contact avec l'environnement. Nous avons également réduit la force de contact (selon la normale au plan XY), ainsi que les oscillations, dues au contact

grâce à l'ajout de la logique floue.

Les trajectoires de translation et d'orientation de l'effecteur, dans l'espace cartésien, sont obtenues par des splines cubiques paramétriques et par des splines cubiques de quaternions.

Une recherche du trou en aveugle, sous un tracé en spirale, a été modélisé puisque le robot n'est pas équipé d'un système de vision. La technique utilise les splines cubiques paramétriques ainsi que les splines de quaternions.

Finalement une brève analyse de la combinaison d'un capteur de forces à la base du robot (utilisé pour estimer les forces et moments à l'effecteur) et du contrôleur d'impédance a été réalisé. Nous avons montré, par simulation, qu'un tel agencement n'est pas favorable, tant dans un environnement terrestre que dans un environnement spatial.

Voici une liste de recherches futurs :

1. Utiliser *ROBOOP* pour contrôler un robot.
2. Faire une validation expérimentale du modéle.
3. Inclure une librairie graphique (ex : compatible avec OpenGL) permettant la détection de collisions dans la tâche d'insertion. Ceci nous permettrait de modéliser des insertions avec des composantes (pièces et trous) beaucoup plus complexes.
4. Simuler le frottement de Coulomb entre l'outil et l'environnement et essayer de valider les inégalitées de coincement (jamming) retrouvées dans [44].

BIBLIOGRAPHIE

[1] C.H. AN, C.G. ATKESON, and J.M. HOLLERBACH. Estimation of inertial parameters of rigid body links of manipulators. In *Proceedings of the IEEE Conference on Decision and Control*, pages 990–995, december 1985.

[2] M.H. ANG and G.B. ANDEEN. Specifying and achieving passive compliance based on manipulator structure. *IEEE Trans. of Robotics and Automation*, 11 :504–515, 1995.

[3] J. ANGELES. *Fundamentals of Robotic Mechanical Systems : Theory, Methods and Algorithms*. Springer-Verlag, 1997. ISBN = 0-387-94540-7.

[4] A.A. BAZERGHI and A.A. GOLDENBERG. Simulation model of insertion process. *Robot8*, pages 6.84–6.96, 1984.

[5] H. BERGHUIS and H. NIJMEIJER. Global regulation of robots using only position measurements. *Systems and Control Letters*, 21 :289–293, 1993.

[6] J.F BROENINK and M.L.J. TIERNEGO. Peg-in-hole assembly using impedance control with a 6 dof robot. In *Simulation in Industry, Proceedings* 8^{th} *European Simulation Symposium*, pages 504–508, oct 1996.

[7] F. CACCAVALE, S. CHIAVERINI, and B. SICILIANO. Second-order kinematic control of robot manipulators with jacobian damped least-squares inverse : Theory and experiments. *IEEE/ASME Transaction on Mechatronics*, 2(3) :188–194, 1997.

[8] F. CACCAVALE, C. NATALE, B. SICILIANO, and L. VILLANI. Resolved-acceleration control of robot manipulators : A critical review with experiments. *Robotica*, 16 :565–573, 1998.

[9] F. CACCAVALE, C. NATALE, B. SICILIANO, and L. VILLANI. Quaternion-based impedance control for dual-robot cooperation. In *9th Intl. Symp. of Robotics Research*, pages 42–49, oct 1999.

[10] F. CACCAVALE and B. SICILIANO. Six-dof impedance control based on angle/axis representations. *IEEE Trans. of Robotics and Automation*, 15 :289–300, 1999.

[11] F. CACCAVALE, B. SICILIANO, and L. VILLANI. Robot impedance control with nondiagonal stiffness. *IEEE Trans. of Robotics and Automation*, 44 :1943–1946, 1999.

[12] S.R. CHHATPAR and M.S. BRANICKY. Search strategies for peg-in-hole assemblies with position uncertainty. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, pages 1465–1470, october 2001.

[13] S. CHIAVERINI and L. SCIAVICCO. The parallel approach to force/position control of robotic manipulator. *IEEE Trans. of Robotics and Automation*, 9(4) :361–373, aug 1993.

[14] S. CHIAVERINI and B. SICILIANO. The unit quaternion : A useful tool for inverse kinematics of robot manipulators. *Systems Analysis, Modeling and Simulation*, 35 :45–60, 1999.

[15] S. CHIAVERINI, B. SICILIANO, and O. EGELAND. Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator.

IEEE Trans on Control Systems Technology, 2(2) :123–134, june 1994.

[16] S. CHIAVERINI, B. SICILIANO, and L. VILLANI. A survey of robot interaction control schemes with experimental comparison. *IEEE/ASME Trans. on Mechatronics*, 4(3) :273–285, sep 1999.

[17] J.C.K. CHOU. Quaternion kinematic and dynamic differential equations. *IEEE Trans. of Robotics and Automation*, 8(1) :53–64, fev 1992.

[18] P. CORKE. A robotics TOOLBOX for MATLAB. *IEEE Robotics & Automation Magazine*, 3(1) :24–32, March 1996.

[19] P. CORKE and B. ARMSTRONG-HÉLOUVRY. A search for consensus among model parameters reported for the puma 560 robot. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1608–1613. may 1994.

[20] J.J. CRAIG. *Introduction to ROBOTICS, Mechanical and Control.* Addison-Wesley Publishing Company, deuxième edition, 1989. ISBN = 0-201-09528-9.

[21] E.B. DAM, M. KOCH, and M. LILLHOLM. Quaternions, interpolation and animation. Technical Report DIKU-TR-98/5, University of Copenhagen, July 1998.

[22] R. DESANTIS. Dynamique des systèmes mécaniques sous contraintes honolomes et non honolomes. Technical report, École Polytechnique de Montréal, 2001. ISBN = 2-553-00908-9.

[23] R.E. ELLIS and S.L. RICKER. Two numerical issues in simulating constrained robot dynamics. *IEEE Trans. on Systems, Man and Cybernetics*, 24(1) :19–27, 1992.

[24] A.C. FANG and B.G. ZIMMERMAN. Digital simulation of rotational kinematics. *Goddard Space Flight Center*, 1(604-31-75-02-51) :1 21, march 1969.

[25] R. GOURDEAU. Object oriented programming for robotic manipulators simulation. *IEEE Robotics and Automation Magazine*, 4(3) :21 29, sep 1997.

[26] R. GOURDEAU, S. BLOUIN, and R. HURTEAU. Computed torque control of robots without joint velocity measurements. In *IEEE Canadian Conf. on Electrical and Computer Engineering*, pages 1413 1418, may 1999.

[27] W.W. HINES and D.C. MONTGOMERY. *Probality and Statistics in Engineering and Management Science*. John Wiley & Sons Inc, troisième edition, 1990. ISBN = 0-471-60090-3.

[28] N. HOGAN. Impedance control : An approach to manipulation, parts i-iii. *ASME J. of Dynamic Systems, Measurement, and Control*, 7 :1 24, 1985.

[29] N. HOGAN. Stable execution of contact tasks using impedance control. *IEEE Trans. of Robotics and Automation*, pages 1047 1054, 1987.

[30] V.C. KLEMA and A.J. LAUB. The singular value decomposition : Its computation and some applications. *IEEE Trans. on Automatic Control*, AC-25(2) :164 176, apr 1980.

[31] J.B KUIPERS. *Quaternions and Rotation Sequences, A Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press, 2002. ISBN = 0-691-05872-5.

[32] D.A LAWRENCE. Impedance control stability properties in common implementations. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages

1185–1190, 1988.

[33] C.C. LEE. Fuzzy logic in control systems : Fuzzy logic controller - part i-ii. *IEEE Trans. on Systems, Man and Cybernetics*, 20(2) :404–435, 1990.

[34] W.-S. LU and Q.-H. MENG. Impedance control with adaptation for robotic manipulations. *IEEE Trans. of Robotics and Automation*, 7(3) :408–415, june 1990.

[35] M.T. MASON. Compliance and force control for computer controlled manipulators. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-11(6) :418–432, june 1981.

[36] G. MOREL and S. DUBOWSKY. The precise control of manipulators with joint friction : A base force/torque sensor method. In *IEEE Intl. Conf. on Robotics and Automation*, pages 360–365, apr 1996.

[37] F. NAGHDY and N.P. NGUYEN. Fuzzy logic compliance control of the peg in hole insertion. *Control Engineering Practice*, 6 :1459–1474, 1998.

[38] Y. NAKAMURA and H. HANAFUSA. Inverse kinematic solutions with singularity robustness for robot manipulator control. *J. of Dynamic Systems, Measurement, and Control*, 108 :163–171, sep 1986.

[39] C. NATALE. Quaternion-based representation of rigid bodies orientation. *PRISMA Technical Report*, 97-05 :1–6, Oct 1997. www.prisma.unina.it/public/frame.htm.

[40] M. NEGNEVITSKY. *Artificial Intelligence, A Guide to Intelligent Systems.* Addison-Wesley Publishing Company, première edition, 2002. ISBN = 0201-71159-1.

[41] S. NICOSIA and P. TOMEI. Robot control by using only joint position measurements. *IEEE Trans. on Automatic Control*, AC-35(9) :1058–1061, sep 1990.

[42] H. BERGHUISand H. NIJMEIJER. Robust control of robots via linear estimated state feedback. *IEEE Trans. on Automatic Control*, 39(10) :2159–2162, 1994.

[43] G.C. PETTINARO. Behaviour-based peg-in-hole. *Robotica*, 17(2) :189–201, 1999.

[44] M. SHAHINPOOR and H. ZOHOOR. Analysis of dynamic insertion type assembly for manufacturing automation. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 2458–2464, april 1991.

[45] K. SHOEMAKE. Animating rotation with quaternions calculus. In *ACM SIGGRAPH*, 1987.

[46] B. SICILIANO and L. VILLANI. Six-degree-of-freedom impedance robot control. In *8th Intl. Conf on Advanced Robotics*, pages 387–392, july 1997.

[47] J.E. SLOTINE and W. LI. *Applied Nonlinear Control*. Prentice Hall, 1991. ISBN = 0-13-040890-5.

[48] B. STROUSTRUP. *The C++ Programming Language*. Addison-Wesley Publishing Company, troisième edition, 1997. ISBN = 0-201-88954-4.

[49] R.H. STURGES JR. A three-dimensional assembly task quantification with application to machine dexterity. *The International Journal of Robotics Research*, 7(4) :34–78, august 1988.

[50] C.W. WAMPLER. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Trans. on Systems, Man and Cybernetics*, 16 :93–101, 1986.

[51] D.E. WHITNEY. Quasi-static assembly of compliantly supported rigid parts. *ASME Jour. of Dynamic Systems, Measurement, and Control*, 104 :65–77, march 1982.

[52] M. WOO, J. NEIDER, T. DAVIS, and D. SHREINER. *OpenGL Programming Guide*. Addison-Wesley Publishing Company, troisième edition, 1999. ISBN = 0-201-604458-2.

[53] J. YEN and R. LANGARI. *Fuzzy Logic, Intelligence, Control and Informatiom*. Prentice Hall, première edition, 1999. ISBN = 0-13-525817-0.

Annexe I

Introduction aux quaternions

Les quaternions peuvent être utilisés pour exprimer une rotation autour d'un axe par un angle. Cette représentation est plus naturelle que les angles d'Euler pour faire de l'animation. Par exemple quels sont les angles d'Euler, représentant des rotations autour des axes de bases, permettant de faire une rotation d'un angle de 30 degrés autour de l'axe $[1, 1, 1]$? De plus l'ordre de rotation des angles d'Euler est important, ce qui fait qu'il existe neuf différentes conventions.

Selon le théorème d'Euler [21], la représentation de rotation par des quaternions est compacte car elle utilise quatre paramètres pour décrire une orientation, contrairement aux angles d'Euler qui utilisent neuf paramètres (élément d'une matrice de rotation). Tous les quaternions non nuls peuvent être utilisés pour représenter des rotations, mais généralement on utilise uniquement les quaternions unitaires. Une rotation peut toujours se représenter par deux quaternions q et $-q$, ce qui semble de prime abord un inconvénient. D'un point de vue mathématique, les quaternions viennent en paire. Une orientation donnée peut être obtenue par deux rotations, de sens opposés, autour d'un axe. Un autre avantage des quaternions est la formulation élégante de l'interpolation des quaternions. Obtenir une courbe continue, par interpolation, est beaucoup plus difficile en utilisant les angles d'Euler. L'interpolation cubique de quaternions présentée à la section 6.2.2 est utilisée pour générer des trajectoires d'orientation.

Il est possible d'implanter des translations au moyen de quaternions. Lors d'une

addition de quaternions, la partie vectorielle du quaternion résultant peut s'interpréter comme un vecteur de translation. On peut même définir un genre de quaternion homogène, dans lequel la multiplication est équivalente aux transformations de rotation et de translation. Cette extension des quaternions n'est pas élégante comparativement aux matrices de transformations homogènes. Les quaternions homogènes ne sont pas vraiment traités dans la littérature, on préfère la matrice de transformation homogène. Les quaternions sont donc utilisés exclusivement pour représenter les rotations et les matrices pour les autres transformations [21].

Voici un résumé de notions importantes portant sur les quaternions unitaires. Pour une référence plus complète, le lecteur devrait consulter une des références suivantes : [31], [21] [17] et [39]. Le livre de Kuipers [31] présente plusieurs illustrations sur l'interprétation des quaternions ainsi que quelques démonstrations simples. Dam, Koch et Lillholm [21] fournissent plusieurs démonstrations mathématiques intéressantes utilisées pour l'interpolation des quaternions.

I.1 Notions générales

L'équation I.1 définit un quaternion

$$q = w + xi + yj + zk \tag{I.1}$$

où w, x, y et z sont des nombres réels. Deux variantes de l'équation I.1 sont présentées ci-dessous :

$$q = [s\ v_1\ v_2\ v_3] \tag{I.2}$$

$$= (s, v) \tag{I.3}$$

où s est un scalaire et v est un vecteur. Les quaternions étant une extension des nombres complexes, le carré des vecteurs i, j et k est de -1. De plus, le produit de deux quaternions se comporte de façon similaire au produit vectoriel du vecteur unitaire.

$$\begin{aligned} ii &= -1 \\ jj &= -1 \\ kk &= -1 \\ ij &= -ji = k \\ jk &= -kj = i \\ ki &= -ik = j \end{aligned}$$

L'addition et la soustraction de deux quaternions sont données par

$$\begin{aligned} q_0 \pm q_1 &= (w_0 + x_0 i + y_0 j + z_0 k) \pm (w_1 + x_1 i + y_1 j + z_1 k) \\ &= (w_0 \pm w_1) + (x_0 \pm x_1) i + (y_0 \pm y_1) j + (z_0 \pm z_1) k \end{aligned} \tag{I.4}$$

La multiplication de deux quaternions est donnée par

$$\begin{aligned} q_0 q_1 &= (w_0 + x_0 i + y_0 j + z_0 k)(w_1 + x_1 i + y_1 j + z_1 k) \\ &= (w_0 w_1 - x_0 x_1 - y_0 y_1 - z_0 z_1) + \\ &\quad (w_0 x_1 + x_0 w_1 + y_0 z_1 - z_0 y_1) i + \\ &\quad (w_0 y_1 - x_0 z_1 + y_0 w_1 + z_0 x_1) j \\ &\quad (w_0 z_1 + x_0 y_1 - y_0 x_1 + z_0 w_1) k \\ &= (s_0 s_1 - v_1 \cdot v_2, s_1 v_2 + s_2 v_1 + v_1 \times v_2) \end{aligned}$$

Le produit de quaternions est associatif mais non commutatif

$$(q_1 q_2) q_3 = q_1 (q_2 q_3) \tag{I.5}$$

$$q_1 q_2 \neq q_2 q_1 \tag{I.6}$$

Le conjugué d'un quaternion est défini comme étant

$$q' = (w + xi + yj + zk)' = w - xi - yj - zk \tag{I.7}$$

Le conjugué du produit de quaternions satisfait les propriétés suivantes :

$$(q')' = q \tag{I.8}$$

$$(q_0 q_1)' = q_0' q_1 \tag{I.9}$$

$$(q_0 + q_1) = q_0' + q_1 \tag{I.10}$$

$$qq' = q'q \tag{I.11}$$

La norme d'un quaternion est obtenue en utilisant son conjugué

$$\|q\| = \sqrt{w^2 + x^2 + y^2 + z^2} = \sqrt{(q'q)} \tag{I.12}$$

La norme d'un produit de quaternion satisfait les deux propriétés

$$\|q'\| = \|q\| \tag{I.13}$$

$$\|q_0 q_1'\| = \|q_0\|\|q_1'\| \tag{I.14}$$

La multiplication inverse d'un quaternion q^{-1} satisfait les propriétés suivantes $qq^{-1} = q^{-1}q = 1$. Elle est obtenue de la façon suivante

$$q^{-1} = \frac{q'}{\|q\|} \tag{I.15}$$

L'inverse fait référence à la multiplication inverse $(1/q)$ et est calculée selon

$$q^{-1} = q'/(qq') \tag{I.16}$$

Si le quaternion est unitaire nous avons

$$1 = s^2 + v_1^2 + v_2^2 + v_3^2 \tag{I.17}$$

I.2 Représentation de rotations

Un quaternion unitaire $q = \cos(\theta/2) + u\sin(\theta/2)$ représente une rotation en 3D d'un vecteur v par un angle θ autour de l'axe u. Le nouveau vecteur (v après avoir subi une rotation) est $R(v) = qvq'$. Le groupe des rotations peut être projeté sur une sphère unitaire, à quatre dimensions, de quaternions unitaires.

L'équation $p' = qpq^{-1}$ est identique à $p' = Rp$, où p est un vecteur et R est la matrice de rotation correspondant au quaternion q. La matrice de rotation R correspondant au quaternion q s'obtient donc facilement [21].

$$\begin{aligned} R(s,v) &= (s^2 - v^T v)I + 2vv^T - 2sS(v) \\ &= \begin{bmatrix} s^2+v_1^2-v_2^2-v_3^2 & 2v_1v_2+2sv_3 & 2v_1v_3-2sv_2 \\ 2v_1v_2-2sv_3 & s^2-v_1^2+v_2^2-v_3^2 & 2v_2v_3+2sv_1 \\ 2v_1v_3+2sv_2 & 2v_2v_3-2sv_1 & s^2-v_1^2-v_2^2+v_3^2 \end{bmatrix} \end{aligned} \tag{I.18}$$

où $S(v)$ est la matrice anti-symétrique de produit vectoriel. Cette matrice est définie

de la façon suivante pour le vecteur u

$$S(u) = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \tag{I.19}$$

Il faut un peu plus de travail pour obtenir le quaternion unitaire q correspondant à la matrice de rotation R. Il suffit d'identifier les termes de la matrice I.18 en utilisant une matrice de rotation R. Le terme scalaire, s, du quaternion est obtenu par $tr(R) = 3s^2 - v_1^2 - v_2^2 - v_3^2$, où $tr(R)$ est la trace de R. De plus, puisque R est une matrice de rotation on a $tr(R) = 1$ [31]. La première composante vectorielle de q, v_1, s'obtient en soustrayant R_{32} par R_{23}, où R_{ii} est l'élément ii de R. Les termes v_2 et v_3 s'obtiennent de façon similaire à v_1. Le quaternion correspondant à une matrice de rotaton est

$$s = \pm\frac{1}{2}\sqrt{R_{11} + R_{22} + R_{33} + 1} \tag{I.20}$$

$$v_1 = \frac{R_{32} - R_{23}}{4s} \tag{I.21}$$

$$v_2 = \frac{R_{13} - R_{31}}{4s} \tag{I.22}$$

$$v_3 = \frac{R_{21} - R_{12}}{4s} \tag{I.23}$$

On remarque que le signe de s est indéterminé. Les signes de v_1, v_2 et v_3 vont changer selon le choix du signe de s. Ceci implique que nous devons choisir entre le quaternion q et le quaternion $-q$. Nous avons choisi d'utiliser un signe positif dans la classe *Quaternion* de *ROBOOP*.

I.3 Dérivée et intégration d'un quaternion

La dérivée d'un quaternion est fournie par l'équation de propagation des quaternions [39]

$$\dot{s} = -\frac{1}{2}v^T w_0 \tag{I.24}$$

$$\dot{v} = \frac{1}{2}E(s,v)w_0 \tag{I.25}$$

$$E = sI - S(v) \tag{I.26}$$

où w_0 est le vecteur de vitesse angulaire dans le repère de base. Si le vecteur est exprimé dans le repère de l'objet, w_b, la dérivée devient

$$\dot{s} = -\frac{1}{2}v^T w_b \tag{I.27}$$

$$\dot{v} = \frac{1}{2}E(s,v)w_b \tag{I.28}$$

$$E = sI + S(v) \tag{I.29}$$

Remarquons que $q = (s, v)$ est toujours le quaternion extrait de la matrice de rotation R_b^0.

I.4 Erreur d'orientation

Pour établir un contrôleur basé sur l'orientation [14], il est primordial de définir l'erreur d'orientation entre les valeurs désirées et les valeurs actuelles [39]. Considérons le repère actuel comme étant R_a^0 et celui désiré R_d^0. Il est possible de définir deux types

d'erreur d'orientation en terme d'une matrice de rotation. Le premier permet de décrire l'erreur d'orientation relative entre les deux repères comme étant

$$R_d^a = R_0^a R_d^0 \tag{I.30}$$

Le quaternion extrait de R_d^a est

$$q_a^{-1} q_d \ = \{s_a s_d + v_a^T v_d, s_a v_d - s_d v_a - S(v_a) v_d\} \tag{I.31}$$

où $S(v_a)$ est définie en I.19. Le second type est défini comme étant

$$R_{d_a} = R_d^0 R_0^a \tag{I.32}$$

Le quaternion qui en est extrait est le même que le cas précédent mais exprimé dans le repère de base.

$$q_d q_a^{-1} = \{s_a s_d + v_a^T v_d, s_a v_d - s_d v_a + S(v_a) v_d\} \tag{I.33}$$

Pour les deux types on peut montrer que $\Delta q = \{1, 0\}$ si et seulement si R_a^0 et R_d^0 coincident. Il est donc suffisant de considérer Δv pour l'erreur d'orientation. Les erreurs d'orientation seront différentes si on utilise q_d ou $-q_d$, même si ces deux quaternions représentent la même orientation. Il est donc préférable de choisir l'angle le plus petit entre q_a et q_d. Pour ce faire on choisit le signe σ de q_d pour que $q_a \sigma q_d \geq 0$. De cette façon on évite les rotations inutiles [21].

I.5 Exponentielle et logarithme

Un quaternion unitaire peut se représenter de la façon suivante : $q = cos\theta + usin\theta$, qui est similaire à un nombre complexe dont la norme est unitaire, $exp(i\theta) = \cos(\theta) + i\sin(\theta)$. L'identité d'Euler pour les nombres complexes peut se généraliser pour les quaternions, $exp(u\theta) = \cos(\theta) + u\sin(\theta)$, où $exp(i\theta)$ est remplacé par $exp(u\theta)$ et $u_i u_i$ est remplacé par -1. Cette même équation représente l'exponentielle du quaternion $q = [0, u\theta]$, où q n'est pas nécessairement un quaternion unitaire. De cette identité, il est possible de définir la puissance et le logarithme d'un quaternion comme étant [21]

$$q^t = (\cos(\theta) + u\sin(\theta))^t = exp(ut\theta) = cos(t\theta) + usin(t\theta) \tag{I.34}$$

$$log(q) = log(\cos(\theta) + usin(\theta)) = log(exp(\hat{u}\theta)) = u\theta \tag{I.35}$$

Le logarithme d'un quaternion unitaire ne donne pas nécessairement un quaternion unitaire.

I.6 Dérivée d'une fonction de quaternion

La dérivée d'une fonction q^t, où q est un quaternion unitaire est

$$\frac{d}{dt}q^t = q^t log(q) \tag{I.36}$$

Si la puissance est une fonction, on obtient

$$\frac{d}{dt}q^{f(t)} = f'(t)q^{f(t)}ln(q) \tag{I.37}$$

Le cas général est lorsque q dépend de t et la puissance est une fonction différentiable de t [21].

$$\frac{d}{dt}(q(t))^{f(t)} = f'(t)(q(t))^{f(t)}ln(q) + f(t)(q(t))^{f(t)-1}q'(t) \tag{I.38}$$

Annexe II

Application des moindres carrés amortis pour l'inversion de la Jacobienne

La Jacobienne d'un robot manipulateur permet de relier le vecteur des vitesses articulaires aux vitesses de l'espace cartésien. L'inversion de la Jacobienne est nécessaire pour certains algorithmes, comme la cinématique inverse en boucle fermée et le contrôle selon la méthode de l'accélération résolue.

$$\nu = \begin{bmatrix} \dot{p} \\ w \end{bmatrix} = J(q)\dot{q} \tag{II.1}$$

L'inverse de la Jacobienne n'est pas définie aux points de singularité de la Jacobienne (lorsqu'elle n'est plus de plein rang). Une méthode pour stabiliser l'inverse d'une matrice (ou pseudo-inverse si la matrice n'est pas carrée) est bien connue dans l'analyse numérique comme étant les moindres carrés amortis (*DLS : Damped Least-Squares*). Cette méthode fut appliquée à la cinématique inverse de robots manipulateurs [38] et [50]. La méthode correspond à résoudre l'équation suivante

$$J^T(q)\nu = \left(J^T(q)J(q) + \lambda^2 I\right)\dot{q} \tag{II.2}$$

où λ est le facteur d'amortissement et I est la matrice identitée. La solution de II.2 est [15]

$$\dot{q} = \left(J^T(q)J(q) + \lambda^2 I\right)^{-1} J^T(q)\nu \tag{II.3}$$

Remarquons que lorsque $\lambda = 0$, II.2 devient identique à II.1. Dans cette méthode il s'agit de minimiser le critère suivant [15]

$$\min_q \left\|\nu - J(q)\dot{q}\right\| + \lambda^2 \left\|\dot{q}\right\|^2 \tag{II.4}$$

qui montre qu'une perte de précision sur q lorsque λ trop grand. Pour une meilleure visualisation, décomposons J selon la méthode des valeurs singulières (*SVD : Singular Value Decomposition*) [30] (*SVD s'applique également sur une matrice non carrée*)

$$J = \sum_{i=1}^{m} \sigma_i u_i v_i^T \tag{II.5}$$

où u_i sont les vecteurs singuliers d'entrées, v_i sont les vecteurs singuliers de sorties et σ_i sont les valeurs singulières ordonnées tel que $\sigma_1 \geq \ \sigma_2 \ldots \geq \sigma_r \geq 0$, avec r comme étant le rang de J. En utilisant II.5, la solution de II.3 peut s'écrire [38]

$$q = \sum_{i=1}^{6} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} v_i u_i^T \nu \tag{II.6}$$

La partie fractionnaire de II.6 se réduit à $\frac{1}{\sigma_i}$ lorsque $\sigma_i \ggg \lambda$. Par contre lorsque la plus petite valeur singulière tend vers 0, la fraction tendra vers $\frac{\sigma_i}{\lambda^2}$. Une faible valeur de

λ procure une solution précise au détriment de la robustesse et inversement. Utiliser une valeur constante de λ sur toute l'enveloppe est inadéquat. On définit une région singulière basée sur la plus petite valeur singulière de J [15].

$$\lambda^2 = \begin{cases} 0 & \text{si } \sigma_6 \geq \epsilon \\ \left(1 - (\frac{\sigma_6}{\epsilon})^2\right)\lambda_{max}^2 & \text{sinon} \end{cases} \tag{II.7}$$

où σ_6 est la plus petite valeur singulière de J, ϵ définit l'étendue de la région singulières et λ_{max} est choisi par l'utilisateur pour obtenir une bonne solution dans la région de singularité. Dans cette étude les valeurs de ϵ et λ_{max} étaient 0.015 et 0.2 respectivement.

Contrairement à [15], nous utilisons une décomposition des valeurs singulières de J au lieu d'approximer les deux plus petites valeurs singulières. Nous avons préféré la précision à la rapidité du calcul.

L'inverse de la Jacobienne selon les moindres carrés amortis a été implantée sous la méthode *jacobian_DLS_inv* de la librairie *ROBOOP* [25].

Annexe III

Introduction à la logique floue

La logique floue n'est pas une logique qui est floue, mais bien une logique qui décrit ce qui manque de netteté. Cette logique est basée sur l'idée que tout possède un degré d'appartenance. Contrairement à la logique boolean qui comporte deux valeurs, la logique floue comporte une multitude de valeurs. Elle décrit le degré d'appartenance, entre 0 (complètement faux) et 1 (complètement vrai), à un ensemble.

Cette annexe résume quelques notions importantes de la logique floue. Pour une lecture plus complète le lecteur pourrait consulter [40], [33] et [53]. Le livre Negnevitsky sur l'intelligence artificielle [40] présente une très bonne introduction sur le sujet. D'un autre côté, Chuen Chien Lee [33] a fourni un excellent résumé de la logique floue utilisée dans le contrôle de systèmes. Chuen Chien Lee a été un étudiant du professeur Lofti A. Zadeh, le fondateur de la logique floue. Finalement le livre Yen et Langari donne un exposé des notions de base jusqu'aux notions avancées.

III.1 Notions de base sur la logique floue

Un ensemble flou (fuzzy set) est défini comme étant un ensemble ayant des frontières floues. Soit X l'univers de discours et x ces éléments. Dans la théorie classique des ensembles, l'ensemble A de X est défini par sa fonction caractéristique, $f_A(x) : X \rightarrow 0, 1,$

où

$$f_A(x) = \begin{cases} 1, & \text{si } x \in A \\ 0, & \text{si } x \notin A \end{cases} \tag{III.1}$$

Selon la logique floue, un sous-ensemble flou A_i de l'univers de discours F est défini par une fonction d'appartenance (tel qu'illustré à la figure III.1), $\mu_A(x) :\rightarrow [0, 1]$, où

$$\mu_{A_i} = max\left\{0; 1 - \frac{x - x_0}{\epsilon}\right\} \tag{III.2}$$

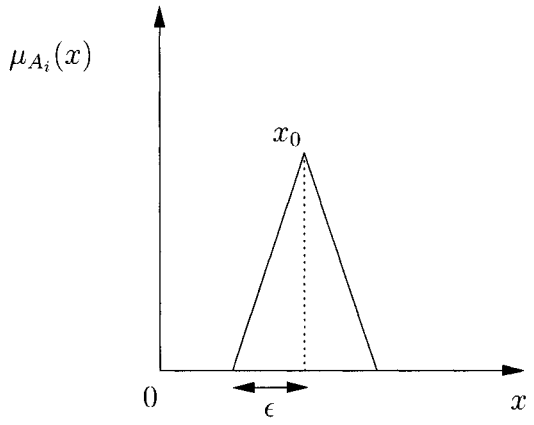


Figure III.1: Fonction d'appartenance du sous-ensemble A_i

III.1.1 Opérateurs et propriétés

Définissons quelques opérateurs flous qui seront utilisés dans les règles d'inférence. Soit A et B deux sous-ensembles flous de l'univers de discours U ayant les fonctions d'appartenance μ_a et μ_b.

Complément : $\overline{\mu}_A(x) = 1 - \mu_A(x)$

Intersection (et) : $\mu_{AB} = min[\mu_A(x), \mu_B(x)]$

Union (ou) : $\mu_{AB} = max[\mu_A(x), \mu_B(x)]$

où $x \in U$.

La logique floue et la logique de Boule, qui peut être considéré comme un cas particulier de la logique floue, possèdent les mêmes propriétés (associativité, commutativité, distributivité, application des règles de De Morgan, etc.).

III.2 Règles floues

Une règle floue peut être définie comme étant une affirmation conditionnelle de la forme :

si x est A

alors y est B

où x et y sont des variables ; et A et B sont des valeurs déterminées par des ensembles flous de l'univers de discours X et Y respectivement.

En général un système flou comportant deux entrées et une sortie peut être décrit

par les règles suivantes :

$$
\begin{aligned}
R_1 : &\quad \text{si } x \text{ est } A_1 \text{ et } y \text{ est } B_1 \text{ alors } z \text{ est } C_1 \\
R_2 : &\quad \text{si } x \text{ est } A_2 \text{ et } y \text{ est } B_2 \text{ alors } z \text{ est } C_2 \\
&\quad \ldots \quad \ldots \quad \ldots \quad \ldots \\
&\quad \ldots \quad \ldots \quad \ldots \quad \ldots \\
R_n : &\quad \text{si } x \text{ est } A_n \text{ et } y \text{ est } B_n \text{ alors } z \text{ est } C_n
\end{aligned}
$$

où x, y et z sont deux variables d'état et une variable de contrôle (sortie) ; A_i, B_i et C_i sont les valeurs des variables x, y et z dans les univers de discours U, V et W respectivement.

Un système de logique floue, noté FLS (Fuzzy Logic System), comprend quatre parties : la fuzzification, l'évaluation des règles floues, l'agrégation de la sortie des règles et la défuzzification. La fuzzification convertit les données discrètes d'entrées vers les ensembles flous d'entrées. Le coeur d'un FLS est sa base de données de règles floues. Elle consiste en un nombre de règles SI-ALORS (ex : R_1 à R_n illustré ci-haut). L'information retrouvée dans ces règles provient essentiellement de la connaissance d'un expert sous la forme linguistique floue (ex : vite, chaud, haut, petit,...) et de valeurs numériques obtenues par le système ou par des capteurs. L'évaluation des règles utilise les règles floues pour produire une relation (mapping) entre les ensembles flous d'entrées et les ensembles flous de sorties. De son côté la défuzzification convertit les ensembles flous de sorties en des valeurs discrètes. Il existe plusieurs méthodes pour

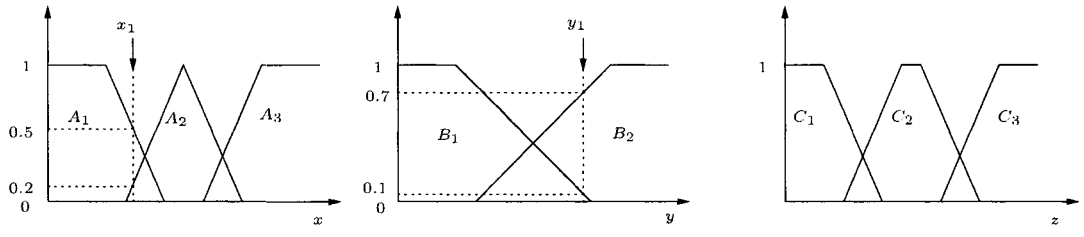
effectuer la défuzzification, la plus populaire est celle du centre de gravité qui a été développée par Mamdani [40]. Les deux principaux avantages de cette méthode sont qu'elle est intuitive et que le temps de calcul en est faible. L'équation III.3 représente la méthode du centre de gravité dans le cas discret.

$$z_1 = \frac{\sum_{i=1}^{n} z_i \cdot \mu_c(z_i)}{\sum_{i=1}^{n} \mu_c(z_i)} \tag{III.3}$$

La principale difficulté dans l'utilisation de la logique floue est d'obtenir de bonnes règles et lois d'inférences. Généralement dans les applications industrielles, les règles et les lois sont obtenues par les connaissances d'un expert pour un système donné.

III.3 Exemple d'application

L'exemple ci-dessous, tiré de [40], illustre les étapes d'un FLS. La figure III.2 illustre la fuzzification des variables d'entrées x_1 et y_1 à partir des ensembles flous A et B respectivement. L'ensemble flou C correspond à la variable de sortie z.

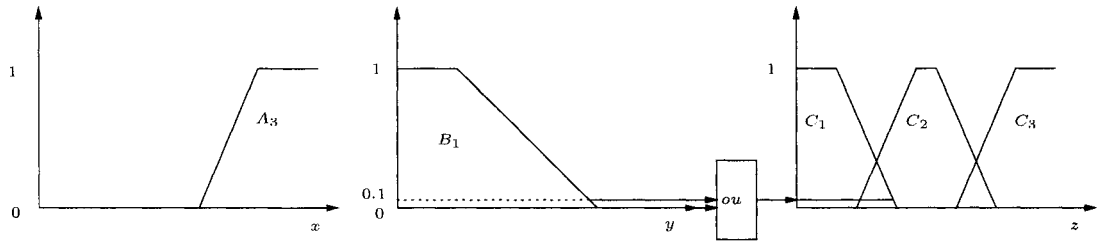


$\mu(x = A_1) = 0.5$ $\quad$ $\mu(y = B_1) = 0.1$
$\mu(x = A_2) = 0.2$ $\quad$ $\mu(y = B_2) = 0.7$

Figure III.2: Fuzzification des variables d'entrées

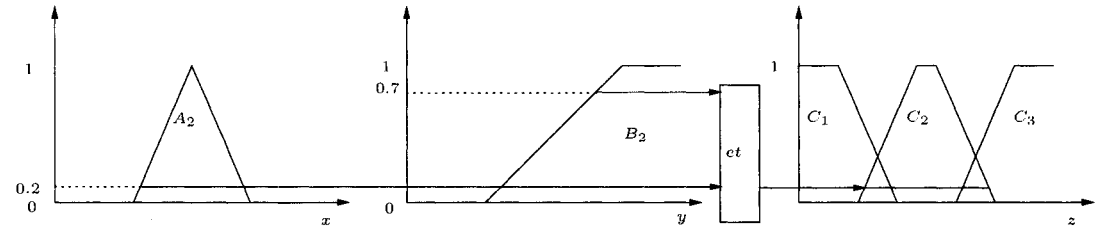
Les figures III.3, III.4 et III.5 illustrent l'évaluation de trois règles floues. De son

côté la figure III.6 présente l'étape de l'agrégation de la sortie des trois règles et la défuzzification basée. Il est à noter que la défuzzification est basée sur la méthode du centre de gravité. La sortie de cette étape est la variable z_1.



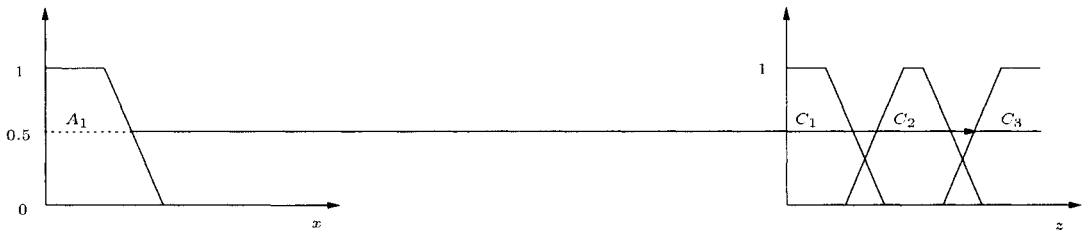
si x est A_3 $(0,0)$ ou y est B_1 (0.1) alors z est C_1 (0.1)

Figure III.3: Évaluation de la première règle



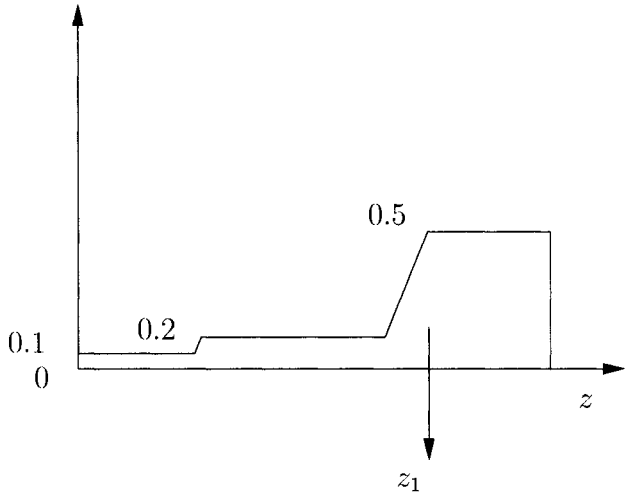
si x est A_2 (0.2) et y est B_2 (0.7) alors z est C_2 (0.2)

Figure III.4: Évaluation de la deuxième règle



si x est A_1 (0.5) alors z est C_3 (0.5)

Figure III.5: Évaluation de la troisième règle



$[z \text{ est } C_1 \ (0.1)] + [z \text{ est } C_2 \ (0.2)] + [z \text{ est } C_3 \ (0.5)]$

Figure III.6: Agrégation des règles et défuzzification

Annexe IV

Introduction à *ROBOOP*

ROBOOP (Robotics Object Oriented Programming) [25] est une librairie écrite en C++ conçue pour la simulation (cinématique et dynamique) de robots manipulateurs. L'incorporation de la librairie *NEWMAT10* permet de traiter les opérations matricielles de façon similaire à Matlab rendant la lecture et la modification du code source plus facile. L'architecture de la librairie utilise la notion d'objet. Ainsi un robot est modélisé comme étant un objet composé d'objets. Afin de bien comprendre le fonctionnement de *ROBOOP* il est impératif d'avoir de bonnes connaissances dans le langage de programmation C++ [48]. *ROBOOP* supporte les conventions Denavit-Hartenberg et Denavit-Hartenberg modifiées pour la modélisation des robots manipulateurs.

La librairie a été conçue à des fins éducationnelles et de recherche, et elle est sous licence GPL. *ROBOOP* peut être téléchargée gratuitement à l'URL suivant : http ://www.cours.polymtl.ca/roboop/download.php.

Les classes fondamentales sont : *Robot_basic*, *Robot*, *mRobot*, *mRobot_min_para*, *Robotgl*, *mRobotgl* et *Link*. Un robot est créé par une instance des classes *Robot* (notation Denavit-Hartenberg), *mRobot* (notation Denavit-Hartenberg modifiée), *mRobot_min_para* (notation Denavit-Hartenberg modifiée et paramètres inertiels minimums), *Robotgl* ou *mRobotgl*. Il est possible d'avoir une animation en *OpenGL* [52] (www.sgi.com/software/ du robot en utilisant une des deux dernières classes. Un robot contiendra autant d'instance de la classe *Link* (membrure) qu'il aura de degrés de liberté. La classe *Robot_basic*

fournit, les propriétés et les méthodes de base communes à toutes les classes de robot.

IV.1 Modifications et ajouts à *ROBOOP*

Les modifications apportées à *ROBOOP* durant ce travail peuvent être consultées à l'URL suivant : http ://www.cours.polymtl.ca/roboop/whatsnew.php. Voici quelques modifications faites par l'auteur de cette étude

1. Des simplifications de l'allocation dynamique de mémoire dans le module de la dynamique a permis de rendre le code plus efficace. Un gain d'environ 10% dans les performances a été mesuré.
2. Correction des équations de la dynamique pour la représentation Denavit- Hartenberg modifiée.
3. Une réorganisation de la hiérarchie des classes a permis de simplifier et de rendre le code plus compréhensible. Cette réorganisation consiste principalement à mettre en place une classe de base, *Robot_basic*, dont les propriétés sont héritées par les classes *Robot*, *mRobot* et *mRobot_min_para*.
4. Ajout d'un affichage graphique réalisé à l'aide de la librairie *OpenGL* pour la visualisation des robots manipulateurs. Les membrures et les joints des robots sont représentés par des cylindres rendant l'affichage générique. La visualisation fonctionne avec les conventions Denavit-Hartenberg et Denavit-Hartenberg modifiée.
5. Ajout de la classe *Quaternion*.
6. Ajout de la classe *Config* permettant de lire des fichiers de configuration. Cette

classe est utilisée par un des constructeurs des classes de robots. Dans un fichier de configuration de robot, on retrouve par exemple l'information sur les paramètres Denavit-Hartenberg et les termes d'inertie. Dans notre projet, la classe *config* a également été utilisée pour initialiser divers paramètres de simulation comme les gains de contrôleurs.

7. Ajout de la classe *IO_Matrix* permettant de lire et écrire des données lors d'une simulation.

8. Ajout des classes *Spl_cubic*, *Spl_path* et *Spl_Quaternion*. La première permet de créer une spline cubique. En utilisant la classe *Spl_cubic*, la classe *Spl_path* génère une trajectoire de position en trois dimensions (X, Y, Z). La classe *Spl_Quaternion* fournit une spline cubique de quaternions, fournissant ainsi une trajectoire d'orientation.

9. Développement du modèle linéarisé pour la convention Denavit-Hartenberg modifiée.

Annexe V

Lexique

Lexique Français/Anglais :

accélération résolue	*resolved acceleration.*
amortis,	*damped.*
amortissement,	*damping.*
anti-symétrique	*skew-symmetric.*
cinématique	*kinematic.*
cinématique inverse	*inverse kinematic.*
coincement	*jamming*
coincement élastique	*wedging*
degré de liberté	*degree-of-freedom*
dynamique	*dynamic.*
dynamique inverse	*inverse dynamic.*
ensemble flou	*fuzzy set*
impédance	*impedance.*
jeu fonctionnel	*assembly clearance*
logique floue	*fuzzy logic*
moindres carrés	*least-squares.*
manipulateur	*manipulator.*
observateur	*observer.*

rigidité	*stiffness.*
splines cubique	*cubic splines*